A Multi-Agent Approach for Hybrid and Dynamic Coevolutionary Genetic Algorithms: Organizational Model and Real-World Problems Applications

Gregoire Danoy

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 482 I.

THESE

présentée par

Grégoire Danoy

Pour obtenir le grade de Docteur
de l'Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialité : Informatique

A Multi-Agent Approach for Hybrid and Dynamic Coevolutionary Genetic Algorithms : Organizational Model and Real-World Problems Applications

Soutenue à Luxembourg le 11 Juin 2008

Membres du jury :

Président :	
Eric Dubois	Professeur, Université de Namur
Rapporteurs :	
Enrique Alba	Professeur, Université de Malaga
Marie-Pierre Gleizes	Professeur, Université de Toulouse
Nikos Vlassis	Maître de conférence, Université Technique de Crète
Examinateurs :	
Eric Dubois	Professeur, Université de Namur
El-Ghazali Talbi	Professeur, Université de Lille
Directeur(s) de thèse :	
Olivier Boissier	Professeur, ENSM.SE, Saint-Etienne
Pascal Bouvry	Professeur, Université du Luxembourg

◘ **Spécialités doctorales :** **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur – Centre G2I
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

◘ **Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat** (titulaires d'un doctorat d'Etat ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
CARRARO	Laurent	PR 1	Mathématiques Appliquées	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1 Professeur 1ère catégorie
PR 2 Professeur 2ème catégorie
MA(MDC) Maître assistant
DR (DR1) Directeur de recherche
Ing. Ingénieur
MR(DR2) Maître de recherche
CR Chargé de recherche
EC Enseignant-chercheur
ICM Ingénieur en chef des mines

Centres :

SMS Sciences des Matériaux et des Structures
SPIN Sciences des Processus Industriels et Naturels
SITE Sciences Information et Technologies pour l'Environnement
G2I Génie Industriel et Informatique
CMP Centre de Microélectronique de Provence
CIS Centre Ingénierie et Santé

Acknowledgements

First of all, I would like to express my sincere gratitude to my two advisors, Pascal Bouvry and Olivier Boissier, whose advice and guiding hand allowed me to conduct my researches in such a motivating and pleasant atmosphere.

I would also like to acknowledge the members of my committee, Enrique Alba, Marie-Pierre Gleizes and Nikos Vlassis for accepting to review this manuscript, El-Ghazali Talbi for being a member of the jury and finally Eric Dubois for being the president of this jury.

I want to congratulate my team mates Luc Hogie, Marcin Seredynski, Riad Aggoune and all the others from Luxembourg and Saint-Etienne. I highly appreciated their considerable advice, support, and influence in addition to their great friendship.

I want to thank the University of Luxembourg for the assistant contract it provided me within the Evo-Business project. It not only allowed me to conduct my research in very good conditions but also to have some teaching experience during those almost four years. Additionally I would like to thank the staff of the University of Luxembourg and of the CSC team for their helpful support.

During this thesis I also had the opportunity to collaborate with people from Luxembourg and abroad whose knowledge and advice greatly influenced and improved my work:

- Enrique Alba, Professor in University of Malaga, who gave me the opportunity to work within his research group and whose experience and help has greatly improved my knowledge and my work quality. Therefore I also want to thank the team of the Department of Computer Science, E.T.S. Ingeniería Informática, for its kindness and and more specifically Bernabe Dorronsoro with whom collaborating has been a real pleasure.

- Franciszek Seredynski, Professor in Polish Japanese Institute of Computer Science and the Polish Academy of Sciences of Warsaw in Poland

- Luc Hogie, Phd, who additionally to being a nice team mate allowed me to improve my knowledge on ad hoc networks and whose simulator and support has been very useful.

- Matthias R. Brust, PhD, whose research on {itshape Injection Networks

- Tommy Martins, Bojan Reljic and Christian Franck, students from the University of Luxembourg, who found some interest in the research works I proposed and helped me extending the DAFO framework as well as conducting some experiments.

On a more personal point of view, I want to highly thank my parents Alain and Martine as well as my three sisters, for supporting me during all these years of studies .

I am also grateful to my family in law whose kindness and presence helped me a lot.

Last but not least, a big thank you to my future wife Stephanie, who shares my life since more than seven years and whose presence means so much to me.

Abstract

Since the mid 1970s and the introduction of Genetic Algorithms (GAs) by John H. Holland, the idea of mimicking the capacity of biological systems to adapt to the genetic level in response to environmental challenges has motivated many research studies for applying similar mechanisms to scientific problems. One recent evolution of such algorithms, namely Coevolutionary Genetic Algorithms (CGAs), focuses on the coevolution of populations (competing or cooperating) of individuals representing specific parts of the global solution instead of evolving a population of similar individuals representing a global solution. This thesis work aims at modeling and applying such CGAs as well as developing new ones in the context of business problems optimization.

In this dissertation we assert that modeling CGAs as organizational multi-agent systems overcomes the lack of explicitness at the level of the algorithms structure, interactions and adaptation to existing models and platforms. We therefore introduce MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, a new agent organizational and re-organizational model based on Moise+ and dedicated to evolutionary optimization. This model was used to describe existing CGAs as well as to build two new variants, hybrid and dynamic, of a competitive CGA.

This thesis also presents the design and implementation of DAFO, a Distributed Agent Framework for Optimization, permitting the use, the manipulation and the distribution of CGAs. Modeled using MAS4EVO and built on top of a multi-agent platform, DAFO allows the application and comparison of various CGAs (existing and novel ones) on optimization problems.

The CGAs experimentations were conducted on two business problems. The first one is an existing inventory management problem for which we studied multiple static instances. We demonstrated the added value of decomposition on small problem instances as well as the improvement brought by the new hybrid and dynamic CGAs. The second problem studied is a new topology control problem in wireless ad hoc networks. State-of-the-art results were obtained while evaluating the performance of different CGAs on multiple static instances and on one dynamic instance of this network optimization problem.

Resumé

Depuis le début des années 1970 et l'introduction des Algorithmes Génétiques (AG) par John H. Holland, l'idée de mimer la capacité d'adaptation au niveau génétique des systèmes biologiques en réponse à des modifications environnementales a motivé de nombreuses recherches utilisant des mécanismes similaires pour des problèmes scientifiques. Une évolution récente de tels algorithmes, appelés Algorithmes Génétiques Coévolutionnaires (AGCs), s'intéresse à la coévolution de populations d'individus (en coopération ou en compétition) représentants des parties spécifiques de la solution globale au lieu d'évoluer une population d'individus similaires représentants la solution globale. Cette thèse a pour objectif de modéliser et d'appliquer de tels AGCs ainsi que d'en développer de nouveaux dans le contexte de l'optimisation de problèmes métier.

Nous défendons la thèse selon laquelle la modélisation des AGCs sous forme de systèmes multi-agent organisationnels répond au manque d'expressivité en terme de structure, d'inter- actions et d'adaptation de ces algorithmes dans les modèles et plateformes existants. Dans cette optique nous introduisons MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, un nouveau modèle agent organisationnel et reorganisationnel basé sur Moise+ et dédié à l'optimisation évolutionnaire. Ce modèle a été utilisé pour décrire et mettre en oeuvre de tels AGCs ainsi que pour construire deux nouvelles variantes, hybride et dynamique, d'un AGC compétitif.

Cette thèse présente également la modélisation et l'implémentation de DAFO (Distributed Agent Framework for Optimization), un framework multi-agent organisationnel permettant l'utilisation, la manipulation et la distribution d'AGCs. Modélisé à l'aide de MAS4EVO et construit sur base d'une plateforme agent existante, il permet d'appliquer et de comparer différents AGCs (existants et nouveaux) sur des problèmes d'optimisation difficiles.

Les expérimentations de ces AGCs ont été conduites sur deux problèmes d'optimisation métier. Le premier est un problème de gestion de stock pour lequel différentes instances statiques ont été étudiées. Nous avons démontré l'apport de la décomposition sur des instances de petite taille ainsi que l'amélioration procurée par les nouveaux AGCs hybrides et dynamiques. Le second problème étudié est un problème de contrôle de topologie dans les réseaux ad hoc sans fil. Des résultats de pointe ont été obtenus lors de l'évaluation des performances de différents AGCs sur de multiples instances statiques et sur une instance dynamique de ce problème d'optimisation de réseaux.

Contents

List of Figures

List of Tables

Resumé Etendu

Introduction

Les travaux présentés dans cete thèse s'inscrivent dans le contexte de l'automatisation de décisions métiers à l'aide d'algorithmes évolutionnaires (AEs).

Les AEs sont des méthodes heuristiques utilisées pour résoudre des problèmes difficiles et sont basées sur les mécanismes de sélection naturelle et la génétique. Concrètement elles appliquent le principe d'évolution de Darwin (la survie du plus fort) parmi des solutions potentielles (dénommées "individus") à l'aide de processus stochastiques de mutation de gènes, de croisement, etc. Les AEs présentent souvent un avantage comparé à de nombreuses méthodes traditionnelles de recherche locale lorsque les espaces de recherches sont discontinus ou hautement contraints et ont par conséquent été appliqués sur de nombreux types de problèmes difficiles allant de l'optimisation statique à l'ordonnancement.

Lorsqu'ils sont appliqués sur des problèmes d'optimisation métier, les objectifs des AEs peuvent être soit de fournir une seule et unique "très bonne" solution (par ex. pour la création d'un réseau routier) soit de fournir de "bonnes" solutions dans un laps de temps très court (par ex. pour la gestion dynamique d'une flotte de véhicules). Nos travaux s'inscrivent dans ce second type de problèmes qui sont typiquement de très haute complexité, dynamiques et décomposés en de multiples sous-problèmes qui doivent être optimisés tout en satisfaisant le l'objectif global. Leur optimisation est par conséquent décentralisée et requiert l'obtention de solution robustes et acceptables à tout moment ainsi qu'en temps-réel (ou tout du moins en un minimum de temps possible). Afin d'appliquer des AEs éfficaces sur ce type de problèmes, ces algorithmes doivent bénéficier de leur decomposabilité, car d'une part il a été prouvé que prendre en compte la structure du problème permet une accélération supra-linéaire et d'autre part prendre en compte ces informations supplémentaires sur le problème peuvent mener à de meilleures solutions.

Les AEs sont une classe d'algorithmes d'optimization qui présente des capacités clés pour l'optimisation de problèmes métier. Malheureusement, les AEs "classiques" ont tendance à obtenir des résultats de moins bonne qualité lorsqu'ils sont appliqués sur certains problèmes, plus spécialement lorsqu'ils présentent de vastes espaces de recherche comme dans les problèmes métier. Ces nouveaux types de problèmes sont intrinsèquement distribués, c'est à dire qu'ils peuvent être vus comme des entités indépendantes qui interagissent, qui possèdent leurs propres buts et dont le comportement global peut

être observé comme le résultats de ces interactions. Dans l'optique de résoudre ce type de problèmes, des chercheurs se sont une nouvelle fois référés à un processus inspiré de la nature afin d'étendre les algorithmes évolutionnaires: la coévolution (c-à-d la coexistence de plusieurs espèces).

Les principales différences entre les Algorithmes Coévolutionnaires (ACs) et les AEs proviennent de la nature adaptative de l'évaluation de la fonction de coût dans les systèmes coévolutionnaires: l'évaluation d'un individu est basée sur ses interactions avec d'autres individus provenant de différentes sous-populations. Par conséquent, au lieu d'évaluer une population d'individus similaires représentants une solution globale tel que dans les AEs classiques, les ACs considèrent la coévolution de sous-population d'individus représentant des parties spécifiques de la solution globale et considèrent donc la décomposition du problème. Intrinsèquement, en faisant coévoluer plusieurs sous-composant les ACs présentent une opportunité pour la parallélisation (par ex. un sous-composant par processeur). Ces systèmes ont été catégorisé en tant que compétitifs ou coopératifs et ont prouvé qu'ils sont une extension populaire des AEs traditionnels pour les problèmes tests (fonction de test, jeux de stratégie, jeux de robots). Cependant les ACs ont encore été peu utilisés pour des problèmes d'optimisation métier.

Afin de faciliter l'utilisation et la comparaison de différents AEs, de nombreuses librairies et plateformes ont été proposées. Cependant, comme nous allons le démontrer dans cette dissertation, seulement quelques unes permettent l'utilisation des ACs. Ceci motiva déjà en partie nos recherches, visant à fournir une nouvelle plateforme dédiée à l'utilisation des ACs. Cependant ceci n'était pas une motivation suffisante pour commencer le développement d'une toute nouvelle plateforme, sachant qu'étendre une des plateforms existantes aurait été suffisant. La seconde raison réside dans la modélisation utilisée dans les plateformes existantes, qui dans la grande majorité est orientée objet, alors que dans la littérature décrivant les ACs leurs sous-populations sont décrites en tant qu'agents. C'est pourquoi notre seconde motivation consiste à bénéficier du domaine multi-agent et de modéliser notre plateforme (et donc les ACs) comme un système multi-agents (SMA). Cette modélisation permettra d'expliciter la structure des ACs (par ex. les interactions entre agents), les dynamiques de cette structure et les interactions avec l'environnement du système (c-à-d le problème d'optimization). De plus, il devient alors possible de bénéficier des plateformes multi-agent existantes afin de faciliter le développement et la distribution de notre framework.

Les principales contributions présentées dans cette dissertation sont les suivantes:

- MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, un nouveau modèle multi-agent dédié à l'optimization évolutionnaire. MAS4EVO apporte une nouvelle approche dans la modélisation et l'implémentation des ACs.

- DAFO, Distributed Agent Framework for Optimization, qui est la plateforme implémentant le modèle MAS4EVO. Basée sur une plateforme agent existante, elle permet, avec peu d'efforts d'implémentation, d'utiliser, de distribuer et de comparer différents ACs sur des problèmes d'optimization tels que des problèmes métiers.

- La création de deux nouvelles variantes d'un AC compétitif, une hybride et une dynamique, et leur intégration dans notre plateforme DAFO. L'étude et la comparaison de leurs performances face à des ACs "standards" sur des problèmes métiers décrits dans le point suivant.
- L'application des ACs (existants et nouveaux) et leur comparaison sur deux problèmes métier. Le premier est un problème de gestion de stock pour lequel différentes instances statiques ont été étudiées. Le second est un nouveau problème de contrôle de topologie dans les réseaux ad hoc. La performance de différents ACs a été étudiée sur différentes instances statiques ainsi que sur une instance dynamique de ce problème d'optimization de réseaux.

Ce résumé étendu est organisé comme suit. Dans la parie État de l'art, nous présentons en premier lieu les algorithmes génétiques coévolutionnaires et en second lieu l'organisation et l'adaptation dans les systèmes multi-agent. La partie MAS4EVO décrit notre nouveau modèle multi-agent dédié à l'optimisation évolutionnaire, ainsi que son utilisation pour modéliser des algorithmes existants et les deux nouveaux algorithmes coévolutionnaires compétitifs hybride et dynamique. Un bref aperçu de son implémentation, DAFO, est également introduit. Dans la partie Experimentations, les résultats obtenus sur deux problèmes d'optimisation métier (gestion de stock et contrôle de topologie dans les réseaux ad hoc) avec les ACs fournis par DAFO sont étudiés. Finalement la section Conclusion présente nos conclusions ainsi que nos perspectives sur les travaux présentés dans ce manuscrit.

État de l'art

Algorithmes Génétiques Coévolutionnaires

Différentes approches existent dans le domaine de l'optimisation. Ainsi, il est possible de distinguer trois grandes classes d'algorithmes d'optimisation: les techniques énumératives (Primal Simplex, Branch&Bound), les approches basées sur le calcul (algorithme glouton, Fibonacci) et les approches stochastiques (recherche tabou, réseaux de neurones, algorithmes évolutionnaires). Dans le cas de l'optimisation de problèmes métiers, la complexité est telle que les méthodes énumératives issues de la recherche opérationnelle ne sont pas adaptées. Les algorithmes basés sur le calcul (de complexité O(n)) procurent rapidement une solution mais généralement trop éloignée de l'optimum global. Il reste donc la classe des algorithmes stochastiques. La plupart des algorithmes de cette classe sont valables, tel que décrit dans le fameux théorème "no free lunch" (il n'y a pas de repas gratuit) [3].

Depuis les années 1940/1950 et l'émergence des ordinateurs modernes, l'idée de mimer certains mécanismes présents dans la nature afin de créer l'Intelligence Artificielle (IA) a captivé de nombreux chercheurs en informatique. De nombreux axes de recherche ont émergé dans la poursuite de ces idées. Un de ces axes s'inscrit dans le domaine de l'informatique et de l'ingénierie et se nomme "calcul évolutionnaire" (CE), qui consiste en l'utilisation de stratégies évolutionnistes pour la résolution de problèmes. Parmi les algorithmes faisant partie du CE se trouvent les Algorithmes Génétiques (AG).

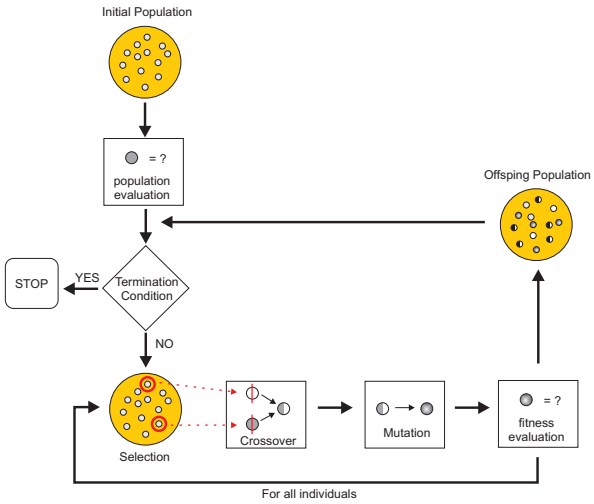


Figure 1: Fonctionnement d'un AG

Les AG furent introduits dans les années 1970 par John H. Holland [4]. Ce dernier examina la capacité d'adaptation au niveau génétique des systèmes biologiques en réponse à des modifications environnementales. Il en résultat les concepts de base de la théorie des AG.

Chronologiquement, les premiers d'AG introduis et également les plus simples sont les AG séquentiels qui peuvent être soit générationnel (generational) [5] soit stationnaire (steady-state) [6].

Un AG commence par générer, généralement aléatoirement, une population d'individus, un individu étant la représentation d'une solution potentielle au problème. Chaque individu est alors évalué sur le problème. Les individus sont ensuite sélectionnés non-déterministiquement en fonction de leur score pour ensuite leur appliquer les opérateurs génétiques de croisement et de mutation afin de générer une nouvelle population. La sélection permet l'exploitation des solutions existantes tandis que le croisement et la mutation permettent d'explorer de nouvelles régions de l'espace de recherche. Cette combinaison entre exploitation et exploration permet à l'algorithme d'évoluer et de converger vers de meilleures solutions. La Figure 1 présente le processus itératif d'un AG.

Depuis lors, les AGs sont devenus une méthode d'optimization populaire et ont été utilisés pour de nombreux problèmes d'optimisation, allant de fonctions de test à des problèmes métiers complexes. Différentes évolutions des AG ont vu le jour, parmi elles nous pouvons citer les AG parallèles (AGP) dont l'objectif est non seulement de réduire le temps de calcul mais aussi de réduire le nombre d'itérations de l'algorithme et d'améliorer la qualité des solutions. Il a en effet été prouvé que les AGP se comportent mieux que la somme des sous-algorithmes les composants, on parle alors d'accélération supra-linéaire. De nombreux modèles d'AGP ont été proposés dans la littérature et de nombreux états de l'art tels que dans [1] [7] et [8] les ont étudié et classifié. Quatre grandes classes regroupent tous ces différents modèles:

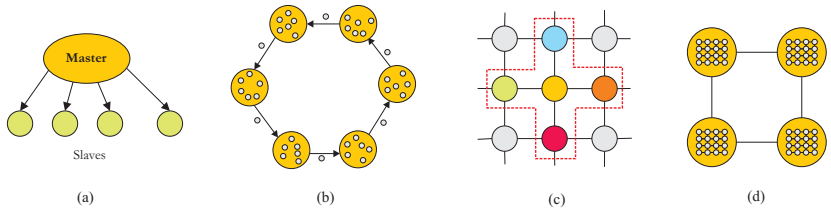


Figure 2: Different Modèles d'AGP: (a) maître-esclave, (b) gros grain, (c) grain fin, (d) hybride

- Le modèle maitre-esclave à population unique (voir Figure 2(a)): représente la façon la plus simple de paralléliser un AG a population unique. Un processeur maître execute un AG réalisant les opérations génétiques de croisement et de mutation. L'operation de l'évaluation de la fonction de coût est parallélisée sur des processeurs esclaves.

- Gros grain (voir Figure 2(b)): les AGP à gros grain consistent en plusieurs sous-populations, chacune executant son propre AG et échangeant périodiquement une partie de leurs individus avec d'autres sous-populations. Cet échange d'individus est appelé migration et est contrôlé par plusieurs paramètres (périodicité des échanges, tailles, etc.). La structure de la population est définie par la topologie du graphe de communication qui spécifie le voisinage de chaque sous-population.

- Grain fin (voir Figure 2(c)): les AGP a grain fin (également appelés modèle cellulaire) n'ont qu'une seule population mais structurée spatialement, cette structure permettant de limiter les interactions entre les individus (un individu ne peut subir d'opération génétique qu'avec ses voisins). Différentes stratégies locales pour appliquer les opérateur génétiques et différents types de voisinages peuvent être utilisés. Ceci permet d'obtenir le même phénomène d'isolation par la distance que l'on retrouve dans le modèle par îlots.

- Hybride (voir Figure 2(d)): les AGP hybrides combinent différents AGP à deux niveaux. La plupart des AGP hybride utilisent des algorithmes à multiples populations au niveau le plus haut. L'exemple présenté en Figure 2 (d) en est une illustration avec un modèle par îlots qui contient un modèle cellulaire dans chacune de ses sous-populations.

Les AGPs ont donc été developpés dans le but de répondre aux limitations des AG séquentiels lorsque appliqués sur des problèmes de grande taille. De nombreuses applications ont été réalisées (voir [1] et [8]) et diverses plateformes permettant leur utilisation ont vu le jour. Cependant, comme pour les AG séquentiels, les AGPs considèrent toujours l'évolution d'individus représentants une solution globale au problème. Ils ne permettent donc pas de décomposition, ni au niveau de la représentation de la solution, ni au niveau de la fonction de coût représentant le problème d'optimisation. La Figure 3 montre un exemple, sur la fonction de Rosenbrock [9], de la représentation d'une solution globale par un chromosome d'un AG.

C'est pourquoi nous nous sommes interessés à une autre classe d'algorithmes évolutionnaires qui permettent de prendre en compte cette décomposition, les algorithmes génétiques coévolutionnaires (AGCs). Les AGCs furent introduit pour la première fois dans les années 1990 [10] et demeurent un domaine de recherche très actif.

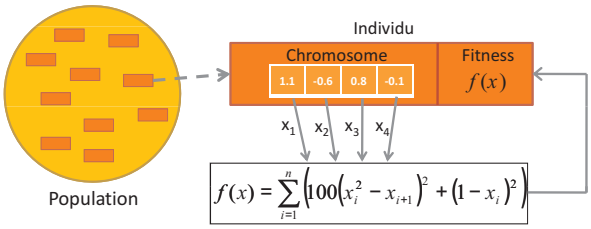


Figure 3: Evaluation d'un individu d'un AG simple pour la fonction de Rosenbrock (n=3)

Le concept de coévolution provient d'observations de la nature qui ont démontré que faire coévoluer plusieurs espèces est plus réaliste que faire évoluer une population unique (globale ou distribuée) contenant les représentants d'une seule espèce. Les individus d'une sous-population représentent alors une partie spécifique de la solution globale au lieu de la représenter dans sa totalité.

Nos travaux nous ont conduit à comparer deux algorithmes génétiques coévolutionnaires, une version coopérative, le Cooperative Coevolutionary GA (CCGA) [11], et une compétitive, le LCGA [12].

Dans CCGA, la décomposition se situe au niveau de la représentation de la solution, en effet chaque espèce représente une partie d'une solution potentielle. Une solution globale est donc obtenue en assemblant un individu de chaque espèce. La nature coopérative de l'algorithme provient du fait que le score d'un individu dépend de la qualité des solutions partielles reçues des autres espèces. L'évolution de chaque espèce est réalisée par un AG indépendant.

La Figure 4 montre l'architecture générale du CCGA (graphe de communication complet) et la façon dont chaque algorithme évolutionnaire calcule le score de ses individus en les combinant avec des représentants séléctionnés parmis les autres espèces (ici avec le meilleur individu provenant de chaque autre espèce).

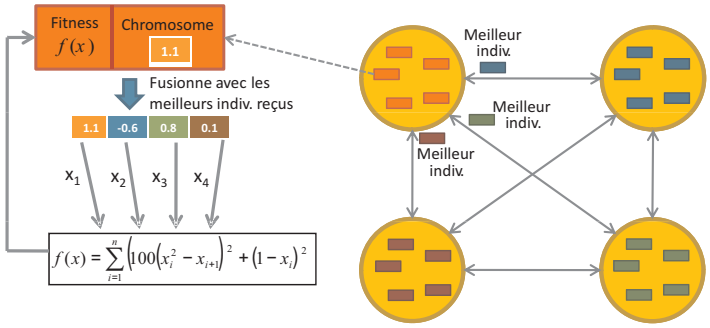


Figure 4: Evaluation d'un individu d'un CCGA pour la fonction de Rosenbrock (n=3)

LCGA est un algorithme coévolutionnaire explorant un paradigme de coévolution compétitive provenant des modèles non-coopératif de la théorie des jeux. Un problème est tout d'abord analysé en fonction de sa possible décomposition et des relations entre ses sous-composants, le tout exprimé par un graphe de communication G_{com} appelé graphe d'interaction. L'objectif est alors de minimiser les

communications entre les joueurs tout en s'assurant que si tous atteignent un optimum local (étant un équilibre de Nash), cela mènera toujours à l'optimum global pour la fonction initiale.

Dans LCGA, chacune des N variables x_i du problème d'optimisation est considérée en tant qu'espèce avec sa propre structure de chromosome, une sous-population étant créée pour chaque variable. Afin d'évaluer la fonction de fitness d'un individu, il est nécessaire de communiquer avec les individus sélectionnés de toutes les autres sous-populations. Le graphe d'interaction est donc dans ce cas complet.

La Figure 5 montre la décomposition du problème utilisée par le LCGA. L'architecture générale du LCGA (ici une liste simple) et la façon dont chaque algorithme évolutionnaire calcule le score de ses individus en les combinant le meilleur individu provenant de l'espèce voisine, dépendent de cette décomposition.

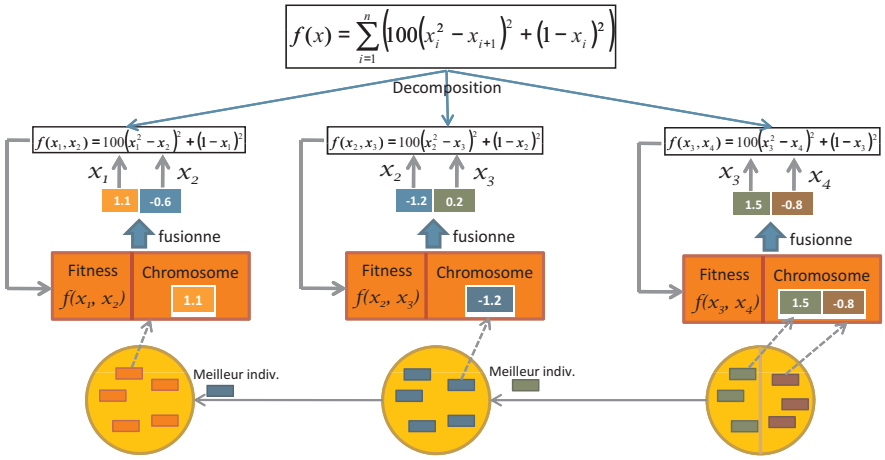


Figure 5: Evaluation d'un individu d'un LCGA pour la fonction de Rosenbrock (n=3)

L'étude des applications de ces AGCs, qu'ils soient compétitifs ou coopératifs, a démontré que les problèmes métiers ont encore été peu traités [13] [14] contrairement aux problèmes tests [15] [16] [11] [17].

Ces deux AGCs diffèrent selon les quatre critères suivants: topologie de communication, mode d'interaction, information échangée et decomposition du problème. Le Tableau 1 présente le CCGA et le LCGA suivant ces quatres

Afin de pouvoir appliquer ces différents AGCs sur des problèmes métier, il est nécessaire de pouvoir spécifier ces caracteristiques à travers un modèle dédié. Ce modèle permettra de manipuler ces paramètres afin d'adapter la structure, les interactions et les informations échangées en fonction du problème d'optimisation.

Afin de faciliter l'utilisation des algorithmes évolutionnaires, de nombreuses plateformes ont été developpées depuis les années 1990. Un état de l'art sur les plateformes les plus populaires telles que ParadisEO [18], MALLBA [19], DREAM [20] ou encore ECJ [21] nous a permis d'observer qu'aucune ne permet l'utilisation de tels algorithmes coévolutionnaires et aucune n'utilise de modèle de haut

niveau pouvant spécifier les quatre critères cités précédemment (topologie, mode d'interaction, information echangée et décomposition). Il s'est donc révélé nécessaire de fournir une nouvelle plateforme permettant l'utilisation des AGCs et facilitant leur application grâce à une spécification basée sur un modèle de haut niveau.

	CCGA	**LCGA**
Topologie	Graph Complet	Pas de restriction
Mode d'interaction	Synchrone	Asynchrone
Information echangée	meilleur individu ou meilleur + aléatoire	Pas de restriction
Décomposition du problème	Non	Possible

Table 1: Comparaison CCGA - LCGA

Organisation et Adaptation dans les Systèmes Multi-Agent

Nous nous sommes donc interessés à l'utilisation du paradigme agent pour notre modèle dédié à l'optimisation évolutionnaire, ce dernier permettant d'exprimer de façon explicite pour l'utilisateur et/ou le système les caractéristiques des différents AGCs.

Afin de décrire notre SMA, nous utiliserons la décomposition introduite par Demazeau dans l'approche voyelle, AEIO, pour Agent, Environment, Interaction et Organisation citeDemazeau95. Cette décomposition procure une concordance simple entre les concepts des AGCs et agents telle que représentée dans le Tableau 2:

AEIO	**AGCs**
Agent	Sous-populations
Environnement	Problème d'optimisation
Interaction	Modes d'interactions + Informations echangées
Organisation	Topologie de communications

Table 2: Concordance AEIO - AGCs

Les *Agents* du SMA représentent les sous-populations des AGCs, l'*Environnement* représente le problème d'optimisation, les *Interactions* représentent à la fois les types d'interactions (synchrones/asynchrones) et leur contenu et finalement l'organisation permet entre autre de structurer ces interactions et donc de définir une topologie de communication.

MAS4EVO

MAS4EVO, Multi-Agent System for Evolutionary Optimisation), est un nouveau modèle multi-agent dédié à l'optimisation évolutionnaire (voir Figure 6). Cette section présente une description détaillée de ce modèle en utilisant la décomposition AEIO telle que présenté dans la section précédente.

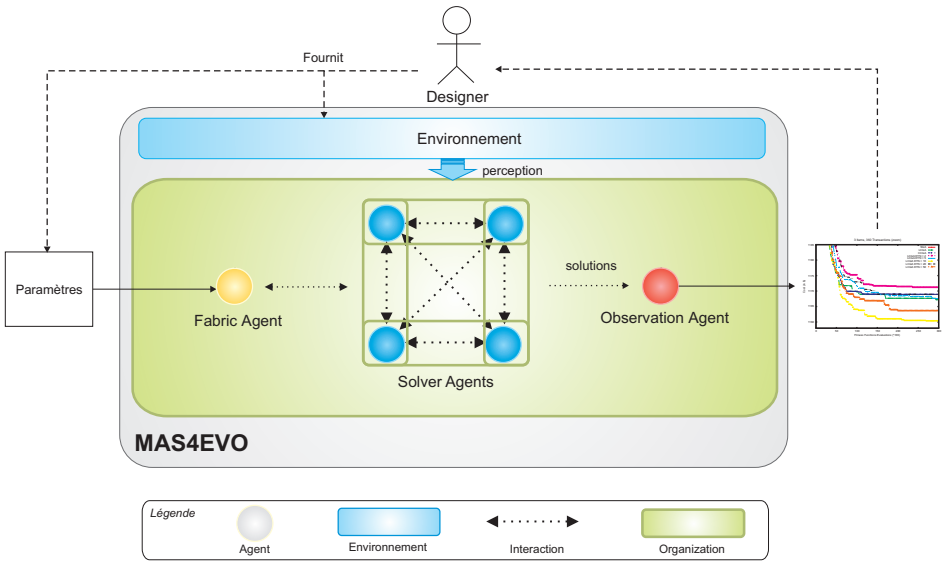


Figure 6: Vue générale de MAS4EVO

Les agents utilisés dans le modèle MAS4EVO possèdent une architecture cognitive telle que présentée en Figure 7. Un agent possède un ensemble de buts à atteindre, un ensemble de compétences lui permettant de réaliser un ou plusieurs buts, ces buts étant contraints par l'Entité Organisationnelle (OE) dans laquelle l'agent st trouve. L'OE est une instantiation d'une Structure Organisationnelle (OS), chaque OS exprimant une stratégie différente correspondant à un AGCs. Un agent sait quels buts sont satisfaits ou non à travers ses états. La fonction de perception permet à un agent d'observer son environnement. La fonction de communication lui permet de dialoguer avec d'autres agents du système, ces interactions étant limitées par l'entité organisationnelle. Finalement, cette architecture comporte un moteur d'inférence qui permet à un agent de selectionner les compétences nécessaires pour réaliser un but.

Le modèle MAS4EVO comprend trois types d'agents qui sont:

- les Agents Fabric: ils ont une vue globale du système et ils gèrent l'initialisation du système en instanciant les agents optimiseurs et en gérant leur cycle de vie.

- les Agents Optimiseurs: ils sont en charge d'optimiser une fonction (mono-objectif) à l'aide d'une métaheuristique (dans notre cas un algorithme génétique ou un algorithme de recherche locale). Ils ont une vue partielle du système dans lequel ils sont situés.

- les Agents Observation: ils ont une vue globale des agents optimiseurs qu'ils monitorent afin d'agreger leurs solutions partielles pour créer la solution globale au problème. Ils procurent une interface avec l'utilisateur grâce aux logs des résultats et aux tracés des courbes correspondantes.

Comme cité précédemment, l'environnement du système représente le problème d'optimisation. Ses caratéristiques sont dépendentes du domaine, le problème pouvant être statique ou dynamique. L'environement est fourni par l'utilisateur, les agents pouvant interagir avec ce dernier grâce à une

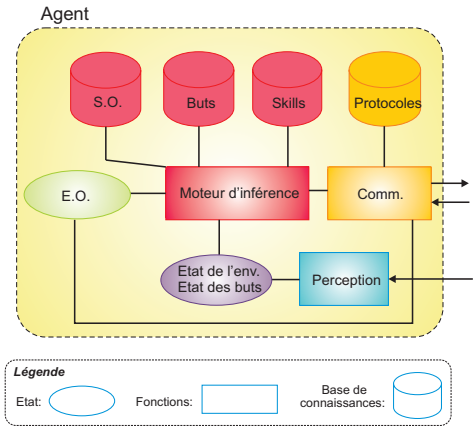


Figure 7: Architecture d'agent

primitive unique, *perception*, qui leur permet d'évaluer leurs solutions sur le problème d'optimisation.

Les interactions au sein du système sont réalisées l'aide d'un langage de communication agent (ACL) compatible FIPA [1]. Ces communications sont structurées sous forme de protocoles d'interactions et utilisent un sous-ensemble des performatifs FIPA (*Perform* et *Agree*). Le langage de communication partagé par les agents de MAS4EVO et les termes le composant ont étés définis et spécifiés à l'aide du langage EBNF [2].

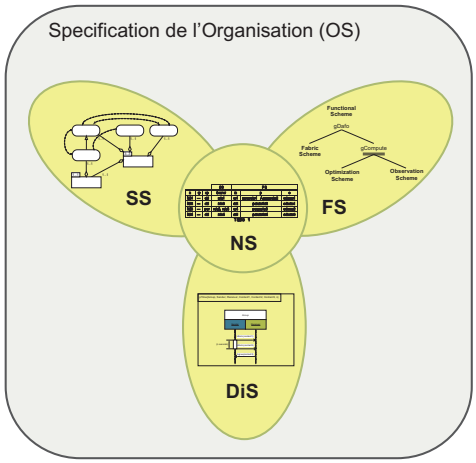


Figure 8: Aperçu du modèle organisationnel de MAS4EVO

Du point de vue organisationnel, MAS4EVO introduit un nouveau modèle organisationnel dédié à l'optimisation évolutionnaire, basé sur Moise+. Ce modèle permet de structurer le fonctionnnement du système et de contraindre les comportements des différents agents qui le compose. Le modèle

[1] FIPA-SL specification: http://www.fipa.org/specs/fipa00008/SC00008I.pdf
[2] Extended Backus-Naur Normal Form

s'articule autour de quatre spécifications:
- Une Spécification Structurelle (SS) définissant les rôles joués par les agents, les relations entre ces rôles et les groupes auxquels ces rôles appartiennent.
- Une Spécification Fonctionnelle (FS) définissant les buts devant être réalisés par l'organisation. Ces buts peuvent être fonctionnels, organisationnels, d'interaction ou de supervision.
- Une Spécification Dialogique (DiS) définissant un ensemble de protocoles d'interactions qui peuvent être utilisés par différents rôles afin de réaliser un ou plusieurs buts d'interaction.
- Une Spécification Normative (NS) définissant les droits et devoirs de chaque rôle ou groupe.

Ces quatre spécifications constituent la spécification organisationnelle (OS) telle que présentée sur la Figure 8.

Nous allons maintenant détailler ces quatres spécifications composant le modèle organisationnel de MAS4EVO et illustrer leur utilisation pour modéliser un algorithme génétique simple (SGA) dans notre plateforme DAFO tel que représenté schématiquement sur la Figure 9.

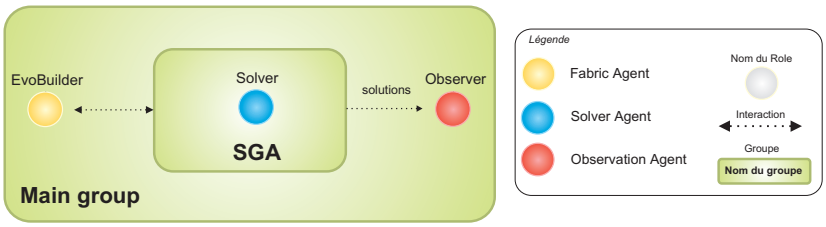


Figure 9: Aperçu d'un SGA modélisé avec MAS4EVO

Spécification Structurelle (SS)

La **Spécification Structurelle** (SS) exprime la structure en terme de rôles, de liens entre rôles et de groupes et peut être représentée graphiquement en respectant un formalisme tel que présenté sur la Figure 10. Un ensemble de contraintes expriment en plus la portée des liens et la cardinalité des rôles et des groupes. Un **rôle** est l'abstraction dun agent dans la SS. Il sert de point d'ancrage d'un agent à un ensemble de contraintes qu'il doit suivre à partir du moment où il accepte de jouer ce rôle. Un **groupe** est l'abstraction dun collectif d'agents. Il est constitué d'un ensemble de rôles, de sous-groupe(s), de liens inter-groupes et intra-groupe et des cardinalités respectivement des rôles et des sous-groupes.

Dans l'exemple de modélisation d'un SGA présenté sur la Figure 10, le groupe racine est le groupe *MainGroup* qui contient un autre groupe *SGA* unique (cardinalité "1..1"). Le groupe *MainGroup* est composé d'exactement trois agents (cardinalité "3..3"), un jouant le rôle *EvoBuilder*, un autre jouant le rôle *Observer* et le dernier jouant le rôle *EvoMember*, chacun de ces rôles ne pouvant être adopté qu'une seule fois (cardinalité "1..1"). Le groupe SGA est composé d'un seul agent (cardinalité "1..1") qui joue le rôle *Solver*, *Solver* héritant du rôle *EvoMember*. Concernant les liens, le rôle *EvoBuilder*

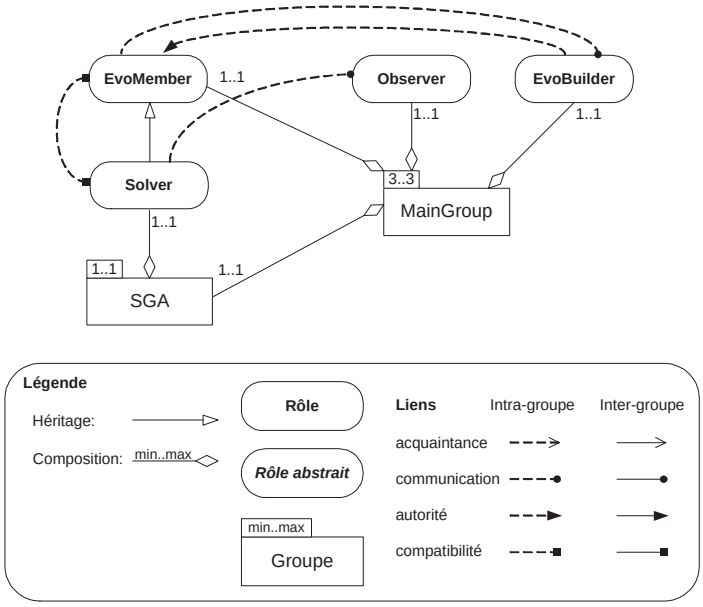


Figure 10: Spécification Structurelle d'un SGA

possède un lien d'autorité sur le rôle *EvoMember* car il contrôle le cycle de vie de ce dernier. Le rôle *EvoMember* possède un lien de communication avec le rôle *EvoBuilder* qui sera utilisé pour communiquer des informations concernant son cycle de vie (par ex. prêt à calculer). Le rôle *Solver* a un lien de communication avec le rôle *Observer* afin de lui communiquer les résultats de ses calculs. Finalement les rôles *EvoMember* et *Solver* ont un lien de compatibilité permettant à un même agent de jouer les deux rôles dans la même instance du groupe *MainGroup*.

Spécification Fonctionnelle (FS)

La **Spécification Fonctionnelle** (FS) définit un ensemble de schémas sociaux considérés comme les buts collectifs à atteindre par l'organisation. Un **schéma social** est la structure d'un objectif global de l'organisation, décomposé en buts structurés en plans et regroupés en missions. Un schéma social peut être représenté par un arbre dont les feuilles sont les buts pouvant être atteints par un agent seul (niveau individuel). Un **but** est un état final constituant un objectif vers lequel lorganisation tend. Un but peut être défini par un plan décomposant le but en sous-buts. Les **missions** regroupent, a priori, les buts en ensembles cohérents, qui devront être accomplis par les agents.

Comparé à Moise+, qui ne permet la définition que de buts fonctionnels, la FS de MAS4EVO permet de définir trois types de buts supplémentaires: organisationnels (impliquant une action sur l'organisation), d'interaction (utilisant un protocole générique de la spécification dialogique) ou de supervision (impliquant l'observation de l'organisation). Une autre extension de MAS4EVO concerne la possibilité d'introduire une contrainte de répétition sur des buts suivant un nombre de répétitions et/ou une contrainte temporelle.

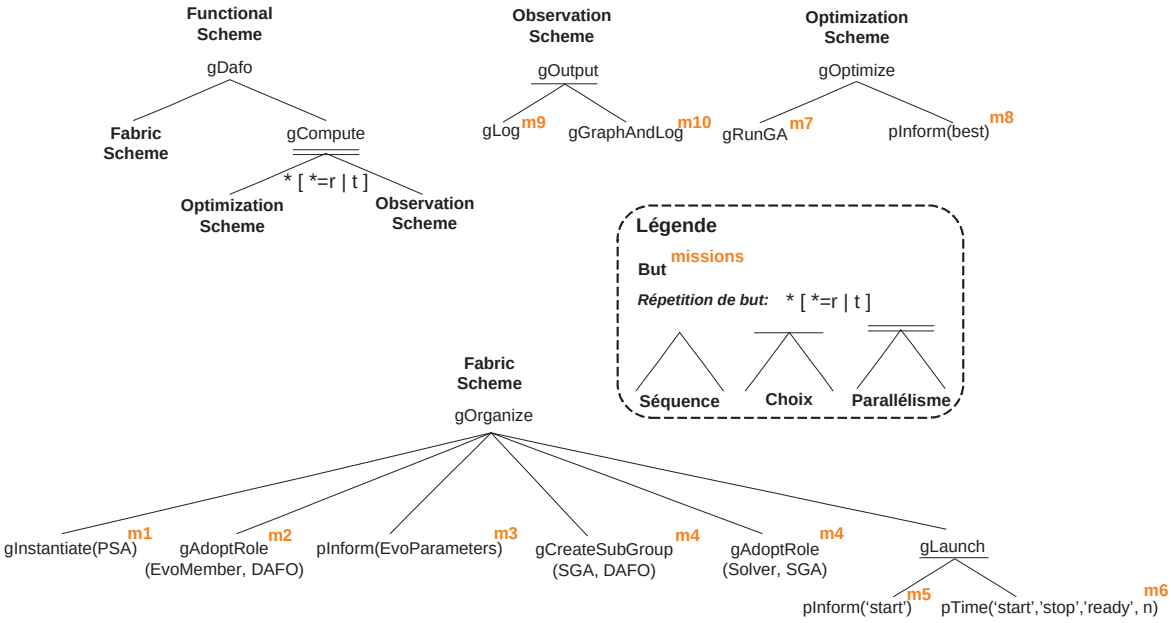


Figure 11: Spécification Fonctionnelle d'un SGA

La Figure 11 présente la FS du SGA qui est basée sur quatre schémas sociaux, *Functional Scheme*, *Fabric Scheme*, *Optimization Scheme* et *Observation Scheme*. Le *Functional Scheme*, considéré comme le schéma principal de la FS, dont le but racine *gDafo* est d'executer le framework, est satisfait lorsque le schéma social *Fabric Scheme* et le but *gCompute* sont séquentiellement satisfaits. Le but *gCompute* est lui-même satisfait lorsque les deux schémas sociaux Optimization Scheme et Observation Scheme sont parallèlement satisfaits.

Spécification Dialogique (DiS)

La **Spécification Dialogique** (DiS) est également un ajout de notre modèle organisationnel par rapport à Moise+. La DiS est principalement basée sur les diagrammes de séquence AUML [22] et sur les diagrammes de séquence organisationnels introduits dans AGR [23]. Elle permet de spécifier des protocoles d'interaction génériques paramètrables indépendamment des rôles et des groupes de la spécification structurelle (SS). La représentation graphique de la DiS a deux dimensions, une dimension verticale représentant le temps et une dimension horizontale représentant les rôles et groupes génériques qui seront spécifiés en tant que paramètres.

La Figure 12 présente deux protocoles d'interaction génériques utilisés pour modéliser le SGA: *pInform* et *pTime*. Le protocole pInform repuiert quatre paramètres pour être instancié, le nom du groupe, le rôle sender, le rôle receiver et le contenu du message. Ce protocole est utilisé dans trois étapes de la spécification fonctionnelle (FS) du SGA (voir Figure 12). Le protocole pTime requiert sept paramètres, le nom du groupe, le rôle sender, le rôle receiver, trois contenus de messages et une

valeur temporelle. Cette dernière valeur représente le temps en secondes entre le premier message Inform et le second.

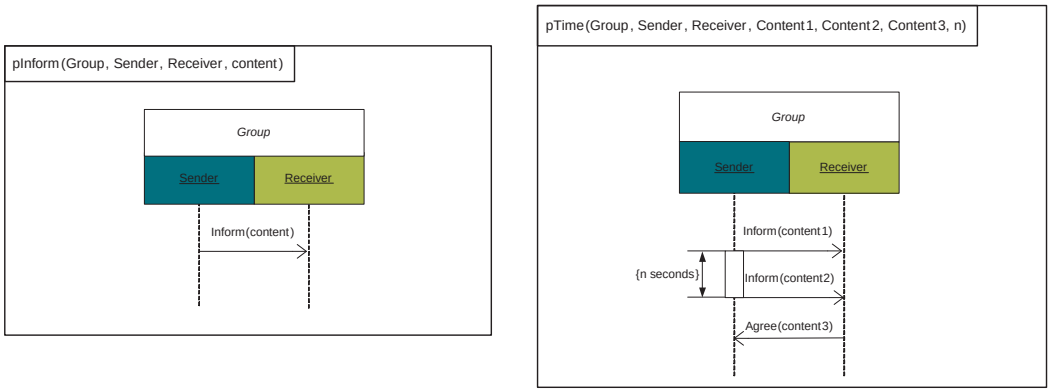


Figure 12: Spécification Dialogique du SGA

Spécification Normative (NS)

Nous venons de décrire la façon dont un ensemble de contraintes pour les agents est défini à l'aide de trois spécifications. Lors de l'exécution de l'Organisation, ils devront ainsi jouer des rôles au sein de groupes tel que spécifié dans la SS, atteindre un ensemble de buts regroupés en missions définis dans la FS et finalement communiquer en instanciant des protocoles d'interaction génériques définis dans la DiS. La **Specification Normative** (NS) permet de lier ces spécifications (tel que représenté sur la Figure 8) à travers la définition de règles concernant un rôle ou un groupe accomplissant une mission composée de buts fonctionnels, organisationnels, de supervision ou d'interaction (instanciant un ou plusieurs protocoles de la DiS). Cette spécification est inspirée de la NS introduite dans Moise-Int [24]

La Spécification Normative est composée d'un ensemble de normes mettant en relation la SS, la FS et la DiS via un porteur, une mission, des paramètres liés à la mission et un opérateur déontique (obligation, permission ou interdiction).

Une norme n est définie selon l'expression suivante: $n = \varphi \rightarrow op(bearer, m, p, s)$ avec:
- φ exprimant les conditions de validité de la norme (la norme reste valide tant que φ est satisfaite);
- op opérateur déontique définissant une Obligation, une Permission ou une Interdiction;
- $bearer$ entité structurelle (un rôle ou un groupe) spécifiée dans la SS, considérée comme la porteuse de la norme;
- m mission spécifiée dans la FS précisant l'action sur laquelle porte la norme;
- p ensemble des paramètres optionnels nécessaires à l'instanciation de la mission m;
- s schéma social de la FS auquel appartient la mission m.

Contrairement aux trois autres spécifications, il est plus difficile de donner une représentation

graphique pour la NS. La représentations utilisée consiste en un tableau contenant chacune des informations définissant une norme. Le Tableau 3 présente la spécification normative du SGA.

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	PSA	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	term==iterations	obl	EvoBuilder, EvoMember	m5	—	Fabric
N06	term==time	obl	EvoBuilder, EvoMember	m6	n	Fabric
N07	true	obl	Solver	m7	—	Optimization
N08	true	obl	Solver, Observer	m8	—	Optimization
N09	mode==batch	obl	Observer	m9	—	Observation
N10	mode==graphical	obl	Observer	m10	—	Observation

Table 3: Spécification Normative du SGA

Nous n'allons décrire textuellement que les normes N01, N05 et N09. La première norme N01 oblige le rôle *EvoBuilder* a accomplir la mission *m1* du schéma social *Fabric*. La norme N05 n'est valide que lorsque la condition de terminaison de l'algorithme est un nombre d'itérations. Dans ce cas elle oblige les rôles *EvoBuilder* et *EvoMember* a réaliser la mission m5 qui contient un but d'interaction. La norme N09 n'est valide que lorsque la plateforme fonctionne en mode console (batch mode). Dans ce cas le rôle Observer est dans l'obligation d'accomplir la mission m9.

Application et Implémentation

MAS4EVO a donc été utilisé pour modéliser des AGCs existants, CCGA et LCGA, ainsi que deux nouvelles variantes du LCGA que nous allons introduire par après, le LCGA hybride (hLCGA) et le LCGA dynamique (dLCGA). Grâce à MAS4EVO, il est possible de modéliser ces différents algorithmes en utilisant un type d'agent générique unique (Problem Solving Agent) et une librairie de modèles organisationnels différents.

Pour des raisons de place, nous n'allons pas présenter l'ensemble des spécifications pour chacun de ces algorithmes, mais uniquement les spécifications démontrant les simples modifications nécessaires pour passer du modèle d'un algorithme à un autre.

CCGA: Spécification Structurelle

En terme de topologie de communication, le CCGA utilise un graphe connecté tel que présenté à gauche de la Figure 13. Afin de modéliser cette topologie en terme de structure organisationnelle, deux nouveaux types de rôles héritant du rôle *Solver* ont été ajouté comparativement à la SS du SGA, *Consumer* et *Producer*. Le rôle *Consumer* joué par un seul *Problem Solving Agent* (cardinalité 1..1) représente la sous-population active alors que le rôle *Producer* est joué par tous les autres PSA (cardinalité All-1). Les agents jouant le rôle de *Producer* peuvent envoyer des informations, généralement

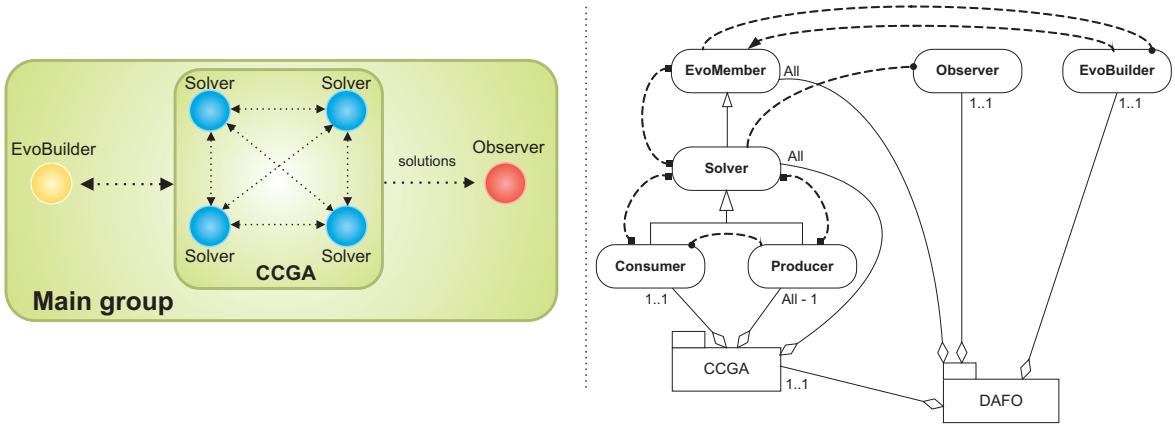


Figure 13: Spécification Structurelle du CCGA

des individus, à l'agent jouant le rôle *Consumer*. Un agent jouant le rôle *Solver* peut également jouer le rôle *Consumer* ou *Producer* (lien de compatibilité).

LCGA: Spécification Structurelle

Comparé au CCGA, afin de modéliser structurellement un LCGA utilisant une topologie de communication en anneau, un nouveau groupe, *Solving Unit*, contenant un seul rôle Producer (cardinalité 1..1) et un à plusieurs rôle Consumer (cardinalité 1..n) a été ajouté (voir Figure 14). Ce groupe *Solving Unit* est un sous-groupe de *LCGA*. Un lien de compatibilité inter-groupe entre les rôles *Producer* et *Consumer* a également été ajouté afin d'exprimer la possibilité pour un agent de jouer ces deux rôles dans deux groupes *Solving Unit* différents.

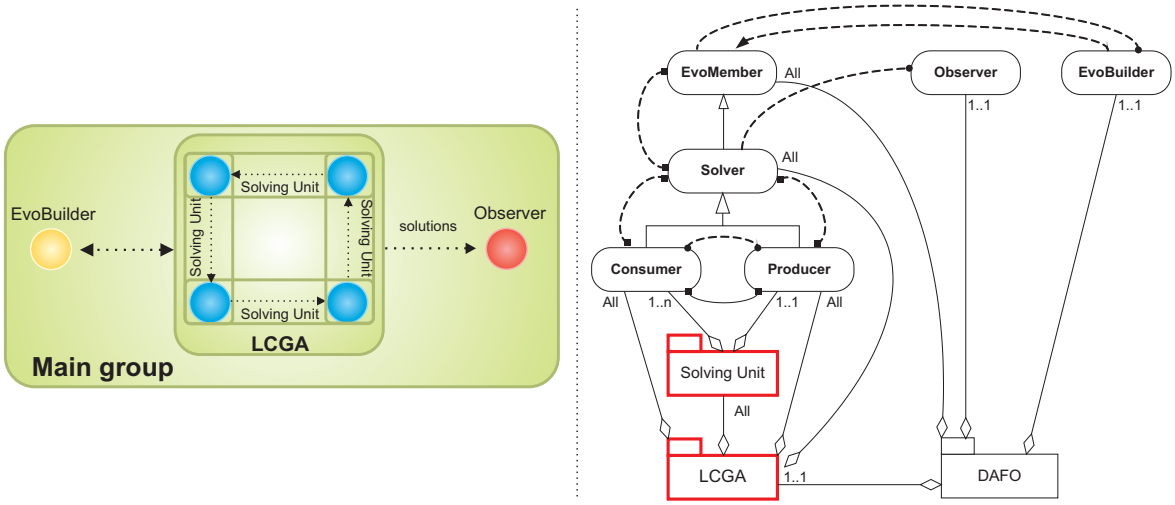


Figure 14: Spécification Structurelle du LCGA

hLCGA: Spécification Structurelle

L'hybridisation des AG a été un domaine de recherche très actif. Cependant, l'hybridisation des AGCs n'a pas fait l'objet d'investigation, hormis les travaux de Son et Baldwick [25] dans lesquels un CCGA est hybridisé avec un algorithme de descente de gradient. Bien qu'il soit possible d'hybridiser les AG de différentes façons, les travaux les plus récents [25] ont démontré que combiner les AG avec des algorithmes de recherche locale sont une méthode efficace pour améliorer leurs résultats. Nous avons donc testé l'hybridisation du LCGA, appelée hLCGA, avec cinq différents algorithmes de recherche locale: Steepest Ascent Hill Climbing (SAHC), Next Ascent Hill Climbing (NAHC), Random Bit Climbing (RBC), Dynamic Hill Climbing (DHC) et Tabu Search (TS).

Ces cinq nouveaux algorithmes ont donc été ajoutés aux compétences des PSA et du point de vue structurel, comparé à un LCGA, un nouveau rôle *LocalSearcher* et un nouveau groupe *LocalSearchUnit* ont été ajoutés (voir partie droite de la Figure 15). Un groupe *LocalSearchUnit* contient exactement deux agents (cardinalité de groupe 2), un jouant le rôle *Solver* (cardinalité 1..1) et un autre le rôle *LocalSearcher* (cardinalité 1..1) ces deux rôles pouvant communiquer entre eux grâce au lien de communication intra-groupe. Ainsi à gauche de la Figure 15 nous pouvons voir une illustration d'un hLCGA où quatre PSA jouent le rôle de Solver dans les groupes *Solving Unit* afin de former un anneau pour le LCGA et ils jouent le même rôle dans les groupes *LocalSearch* Unit où quatre autres PSA jouent les rôles de *LocalSearcher* afin d'hybridiser ce LCGA.

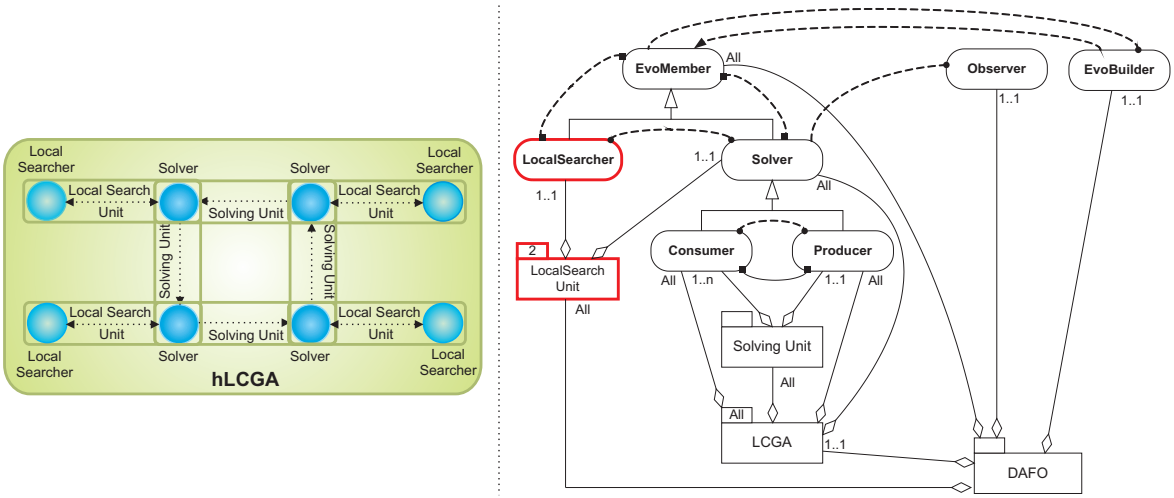


Figure 15: Spécification Structurelle du hLCGA

dLCGA: Spécification Fonctionnelle

Comme pour l'hybridisation, peu de travaux ont étudié la possibilité d'adapter dynamiquement les AGCs. Les quelques travaux relatifs à ce domaine de recherche ont investigé soit l'adaptation du nombre de sous-populations [26] soit l'adaptation de paramètres propres aux AG (croisement, mutation, etc.) [27][28]. Une autre contribution de nos travaux a consisté à créer une version dynamique du

LCGA, denommée dLCGA, dans laquelle la topologie de communication évolue au cours de l'execution. En effet, contrairement au CCGA où la topologie est fixe (graphe complet), le LCGA n'impose aucune restriction quant au graphe utilisé, sachant qu'il dépend de la décomposition de la fonction de coût. Cette réorganisation est effectuée après un nombre prédéfini d'itérations n de l'algorithme. A la gauche de la Figure 16 se trouve une illustration du fonctionnement de dLCGA utilisant une topologie en anneau. Au cours de la première phase, *monitoring*, chaque PSA observe si le critère de réorganisation est atteind, c'est à dire si le nombre d'itération n est réalisé. Une fois le critère satisfait par tous les PSA, ces derniers quittent leurs groupes Solving Unit et commencent la deuxième phase, *négociation*, afin de déterminer leur nouvel emplacement sur l'anneau. Vient ensuite la troisième phase, *réorganisation*, dans laquelle les PSA prennent leurs rôles dans les groupes nouvellement choisis.

Comme le changement de topologie consiste à déplacer les PSA sur l'anneau, ceci n'affecte que l'entité organisationnelle (OE) et non pas la structure organisationnelle (OS). Il n'y a donc aucune différence du point de vue SS entre le LCGA et le dLCGA, c'est pourquoi nous présentons ici les différences au niveau de la FS dont le schéma principal *Functional Scheme* est illusté à droite de la Figure 16. Seul un nouveau schéma social *Reorganization Scheme* a été ajouté à la FS du LCGA, ceci afin de gérer l'ensemble des trois phases (monitoring, negociation et réorganisation) décrites précédemment. Un nouveau but gDynamic a été ajouté afin au shéma principal *Functionnal Scheme* afin de permettre l'ajout de ce nouveau schéma social. Ce schéma est donc executé en séquence aprés le shéma d'optimisation si le critère de monitoring (n générations) est verifié.

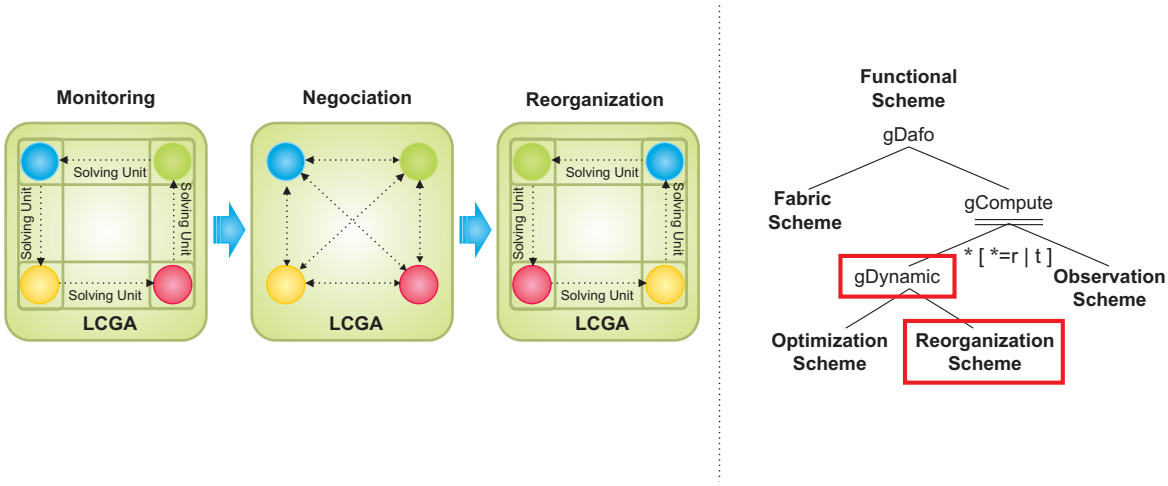


Figure 16: Spécification Fonctionnelle du dLCGA

Implémentation

Nous venons de présenter le modèle MAS4EVO et son utilisation pour modéliser deux AGCs existants (CCGA et LCGA) et deux nouvelles variantes du LCGA (dynamique et hybride). Afin de pouvoir appliquer ces algorithmes et bénéficier du modèle MAS4EVO, ce dernier à été implémenté sous forme

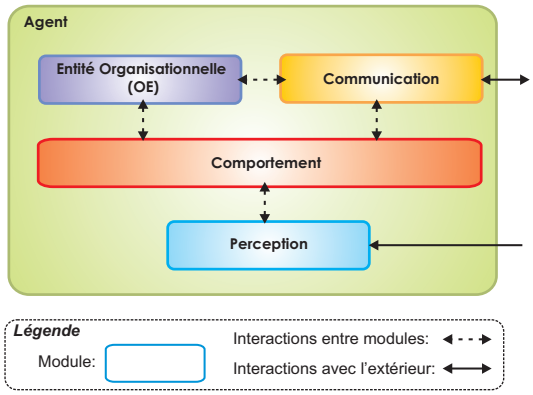


Figure 17: Architecture Modulaire de DAFO

d'une plateforme denommée DAFO (Distributed Agent Framework for Optimization). Cette implémentation est basée sur la plateforme agent Madkit [29] ce qui permet de simplfier le code au niveau des agents, de la structure organisationnelle (Madkit est basé sur le modèle organisationnel AGR pouvant être assimilé à une simplification de notre spécification structurelle), de la communication et de la distribution. De plus Madkit présente de bonnes performances comparé à d'autres plateformes reconnues [30].

Les agents de DAFO sont implémentés de façon modulaire tel que présenté dans la Figure 17.
- Le module **Agent** représente les trois différents types d'agents (PSA, OA et FA). Il gère le cycle de vie de l'agent ainsi que l'activation des autres modules qui le compose.
- Le module **Entité Organisationnelle (OE)** représente l'instanciation des quatres spécifications du modèle organisationnel de MAS4EVO (structurelle, fonctionnelle, dialogique et normative).
- Le module **Communication** implémente les méchanismes d'interaction et plus particulièrement les protocoles d'interaction définis dans la spécification dialogique.
- Le module **Comportements** contient un ensemble de comportements, à la façon des behaviors de la plateforme Jade [31]. En fonction des missions qu'il a à accomplir, un agent activera un ou plusieurs comportements, un comportement mettant en oeuvre des capacités de l'agent afin de réaliser les buts correspondants à ces missions. Ce module est lié au module OE car il est en charge d'adopter le(s) rôle(s) dans le(s) groupe(s) de l'instance de la SS ainsi que de satisfaire les missions et les buts définis dans l'instance de la FS. Le module comportement est aussi lié au module Communication car certaines missions de la FS peuvent inclure des buts d'interaction. Dans ce cas les protocoles d'interaction inclus dans le module communication seront utilisés.
- Le module **Perception** permet à l'agent de percevoir son environnement, c'est à dire les autres agents du système et l'environnement du SMA (qui dans notre cas est le problème d'optimisation fourni par l'utilisateur).

L'optimisation d'un problème à l'aide d'un des algorithmes modélisé avec MAS4EVO et implémenté dans DAFO requiert la définition d'un fichier de configuration basé sur un langage de description dédié: DAFODL (DAFO Description Language). L'objectif est de procurer une description suffisament simple

et complète, à la fois pour les utilisateurs et pour la plateforme elle-même. Un fichier de configuration DAFODL utilise la syntaxe XML et doit respecter une DTD spécifiant la strucutre et le contenu possibles. Ce fichier est divisé en deux parties principales, la première décrivant l'organisation d'agents et la seconde décrivant les paramètres des algoritmes génétiques. Une troisième partie optionnelle permet de décrire les paramètres des algorithmes de recherche locale dans le cas d'un algorithme hybride. Le Tableau 4 présente un exemple de fichier de configuration DAFODL pour un hLCGA utilisant un algorithme de recherche tabou.

```
<?xml version='1.0' encoding='UTF-8'?>
<evoframework>
    <organisation>
        <topology>Ring</topology>
        <numberofagents>10</numberofagents>
    </organisation>
    <geneticparameters>
        <algorithm>hLCGA</algorithm>
        <fitnessclass>MadhocFitnessCalculator</fitnessclass>
        <experiments>20</experiments>
        <terminationcondition>Time</terminationcondition>
        <terminationconditionvalue>60000</terminationconditionvalue>
        <numchroms>100</numchroms>
        <numgenes>2</numgenes>
        <sizegenes>16</sizegenes>
        <crossRate>0.8</crossRate>
        <mutrate>0.03</mutrate>
        <elitenumber>1</elitenumber>
    </geneticparameters>
    <localsearchparameters>
        <lsalgorithm>TabuSearch</lsalgorithm>
        <lsexchangedinformation>PopulationRate</lsexchangedinformation>
        <lspopulationrate>0.01</lspopulationrate>
        <lsterminationCondition>Restricted</lsterminationCondition>
    </localsearchparameters>
</evoframework>
```

Table 4: Exemple d'un fichier de configuration DAFODL

Cette description sera parsée par le *Fabric Agent* afin d'instancier l'organisation de Solver Agents correspondante tel que présenté en Figure 6. Ensuite les paramètres liés aux algorithmes génétiques et de recherche locale seront transmis à ces agents afin qu'ils démarrent leurs calculs.

Experimentation

Dans les sections suivantes nous présentons deux cas d'utilisation de DAFO pour l'optimisation de problèmes métiers. Le premier consiste en l'extension de travaux existant dans lesquels un CCGA a été utilisé pour optimiser un problème de gestion de stock utilisant un modèle à point de commande et le second est un nouveau problème de contrôle de topologie dans les réseaux ad hoc hybrides. L'objectif est d'une part de valider la modélisation des différents CGAs avec MAS4EVO et leur implémentation dans MAS4EVO et d'autre part d'évaluer les performances des nouveaux LCGAs (hybride et dynamique).

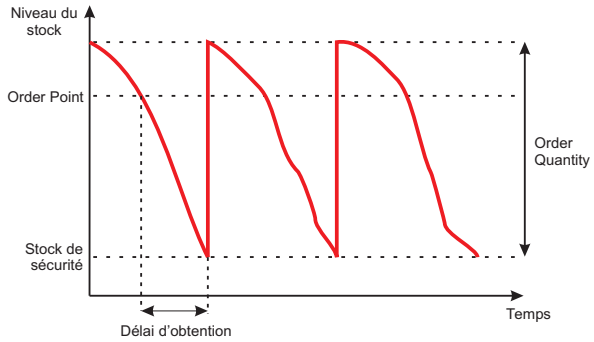


Figure 18: Modèle de gestion de stock à point de commande

Gestion de Stock

Le problème d'optimisation de gestion de stock est décrit en détail dans [14], où Eriksson y démontre les meilleures performances du CCGA comparé à un algorithme génétique standard (SGA). L'objectif est de définir, pour chaque type de produit présent dans le stock, le couple considéré comme fixe Order Point (OP)/Order Quantity (OQ) correspondant à quand et combien commander, afin de minimiser le coût total lié à la gestion du stock. Dès que l'on atteind l'OP, qui est composé de la demande attendue plus un stock de securité, une commande est passée pour une certaine OQ (voir Figure 18. Le coût total est la somme de différents coûts incrémentés lors de chaque transaction (lost sales costs, transportation costs, storage space costs and order costs). Une transaction est composée d'un numéro de transaction, d'une date, d'une référence correspondant au type d'objet demandé et d'une quantité. Lors du traitement de chaque transaction, des actions et décisions sont prises en fonction des paramètres de contrôle du stock (ICP, Inventory Control Parameter) et du niveau du stock. Ces transactions sont executées une à une, dans l'ordre chronologique.

Nous avons donc étendu les travaux d'Eriksson en comparant les performances de nos différents algorithmes, LCGA, dLCGA et hLCGA à celles du CCGA.

Pour chacun de ces algorithmes, une solution à ce problème d'optimisation est encodée sous forme d'un chromosome binaire où chaque gène représente un paramètre du problème, c'est à dire un point de commande (OP) ou une quantité de commande (OQ). Chaque gène est encodé sur 16 bits. Dans le cas du SGA et du CCGA un individu représente les paramètres pour l'ensemble des types de produits du stock. Par exemple, si le stock contient 10 types de produits, chacun nécessitant un OP et un OQ, l'individu possède donc 10*2 = 20 gènes. Pour le LCGA et ses deux variantes (hLCGA et dLCGA), le problème global a été décomposé de telle sorte que chaque sous-population optimise les valeurs d'OP et d'OQ pour un type de produit en fonction d'un deuxième type de produit. Une topoogie de communication en anneau a donc été utilisée telle que présentée en Figure 19.

Différents scénarios de ce problème d'optimisation on été traités, avec 3, 10 et 100 types de produits et respectivement 360, 1200 et 12000 transactions. Nous présentons ici les résultats obtenus avec le SGA, CCGA, LCGA et dLCGA sur l'instance du problème avec 3 produits et 360 transactions. Afin de comparer les performances de ces différents algorithmes, les paramètres suivant ont été utilisés:

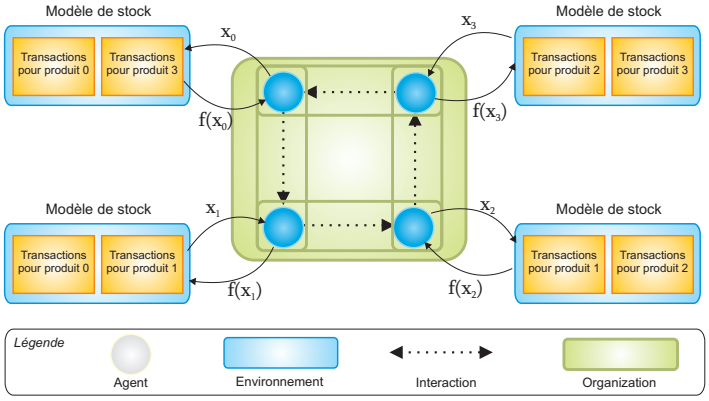


Figure 19: Décomposition utilisée pour LCGA, hLCGA et dLCGA

taille de(s) populations(s) de 100 individus, probabilité de croisement p_c=0.6, probabilité de mutation p_m=1/taille_du_chromosome. Les résultats présentés ci-après sont des moyennes obtenues sur 25 expérimentations indépendantes.

Nombre de sous-populations	3
Taille de Population	100 individuals
Condition de terminaison	30,000 évaluations de fonction
Séléction	Tournoi binaire
Opérateur de croisement	Uniforme, p_c=0.6
Opérateur de mutation	bit flip, p_m = 1/taille_du_chromosome
Elitisme	1 individu

Table 5: Paramètres utilisés pour le SGA, CCGA, LCGA et dLCGA

Il apparait clairement sur la Figure 20 que les AGCs ont de meilleures performances que le SGA à la fois en terme de vitesse de convergence et de meilleur résultat obtenu (coût minimum). Comparé au CCGA, le LCGA converge légèrement plus vite et atteind un résultat légèrement meilleur (voir Tableau 6). Finalement, le dLCGA avec deux intervalles de réorganisation différents (réorganisation toutes les 10 ou 50 générations) permet d'améliorer les résultats obtenus avec le LCGA, aussi bien en terme de vitesse de convergence qu'en terme de meilleurs résultat, le meilleur résultat étant obtenu avec une réorganisation toutes les 10 générations.

Cette première application du LCGA et de ses deux variantes sur le problème métier ICP, déjà optimisé par Eriksson avec le CCGA, a permis de valider le modèle MAS4EVO et son implémentation (DAFO). Nous avons également démontré experimentalement qu'en utilisant une décomposition du problème global, le LCGA et ses deux variantes obtiennent de meilleurs résutltas que le CCGA optimisant le problème global, ceci sur une instance de petite taille du problème ICP. Finalement nous avons également démontré l'amélioration des performances apportée par le hLCGA et le dLCGA sur le LCGA "standard".

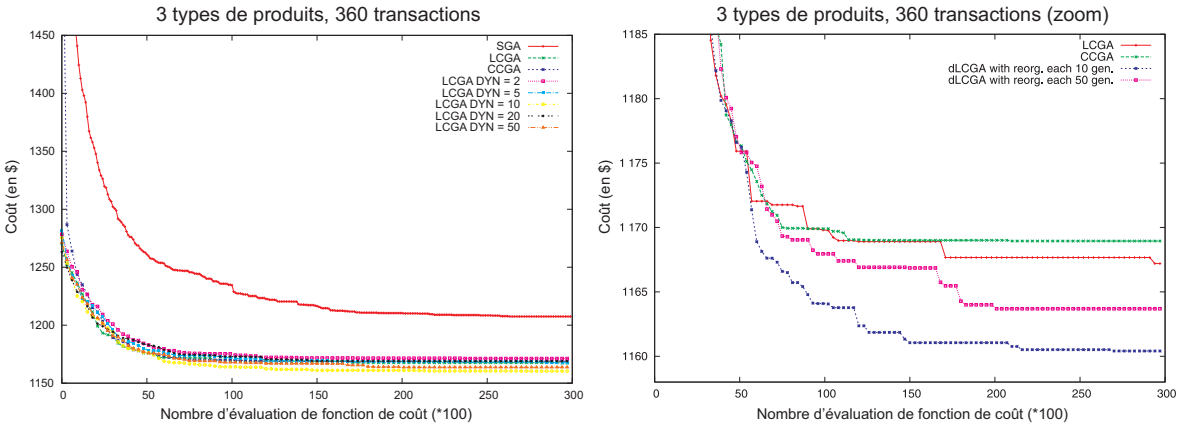


Figure 20: Optimisation du problème ICP (3 types de produits et 360 transactions) avec le SGA, CCGA, LCGA et dLCGA

ICP Parameters	Algorithm	Result
3 Items, 360 Transactions	SGA	1207.47
	CCGA	1168.95
	LCGA	1167.20
	dLCGA, step = 10	1160.41
	dLCGA, step = 50	1163.69

Table 6: Résultats pour le SGA, CCGA, LCGA and dLCGA sur le problème ICP

Réseaux d'Injection (Injection Networks)

Le second problème métier que nous avons optimisé à l'aide de DAFO est un problème de contrôle de topologie dans les réseaux ad hoc hybrides appelé *Réseaux d'Injection (Injection Networks)*. Les réseaux ad hoc sont des réseaux constitués de noeuds capables de s'interconnecter spontanément sans aucune infrastructure préexistante. Les noeuds situés dans leur zone de couverture respectives se connectent en point-à-point. En raison du rayon de couverture limité des noeuds, ces réseaux rencontrent des problèmes de partitionnement qui pénalisent leurs performances globales. Notre problème d'optimisation consiste à placer des liens longue distance (GSM, UMTS ou HSPDA), dénommés *Bypass Links*, afin d'interconnecter ces différentes partitions. Deux noeuds connectés via un *Bypass Link* sont appelés *Injection Points* (voir Figure 21).

Afin d'optimiser le nombre et le placement des ces liens, nous nous sommes intéressées aux propriétés des réseaux petit monde (small world networks). Les réseaux petit monde sont une classe de graphes aléatoires qui présentent une distance géodésique moyenne L (characteristic path length) proche de celle d'un graphe aléatoire ($L \approx L_{random}$) et un fort coefficient de clustering γ (clustering coefficient) largement supérieur ($\gamma \gg \gamma_{random}$). L'utilisation des réseaux petits monde est motivée par le fait qu'ils combinent les avantages des réseaux aléatoires (faible distance géodésique moyenne) et des réseaux réguliers (coefficient de clustering élevé).

L'objectif de ce nouveau problème d'optimisation est donc:

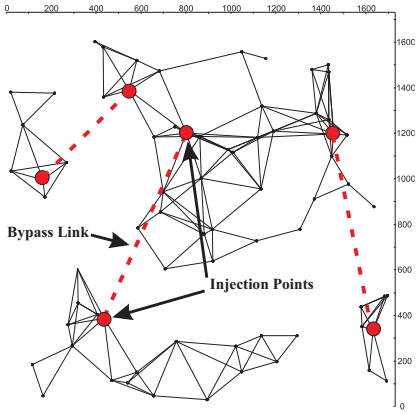


Figure 21: Exemple d'un réseau d'injection

- de minimiser le nombre de partitions
- de minimiser le nombre de bypass links
- de maximiser le coefficient de clustering
- de minimiser la distance géodésique moyenne.

Afin d'optimiser les propriétés petit monde de ces réseaux hybrides, nous avons utilisé un simulateur de réseaux ad hoc: Madhoc [32]. L'emploi de ce simulateur a été motivé par la possibilité de simuler des réseaux hybrides, ses modes de fonctionnement console ou graphique, ce dernier permettant de faciliter la compréhension des différentes alternatives de topologies.

Pour assigner un poids aux solutions potentielles de nos algorithmes, nous avons défini une fonction d'évaluation F modélisant ce nouveau problème de contrôle de topologie. Cette fonction est une combinaison linéaire des deux mesures des réseaux petit monde (L et γ) et du nombre de bypass links créés.

Algorithme 1 : Fonction d'évaluation

si *Graphe connecté* **alors**
  F = α * γ - β * (L - 1) - δ * (bl - bl_{max})
fin
sinon
  fitness = ξ * P
fin

α, β, δ et ξ sont des poids définis experimentallement, bl est le nombre de bypass links créés par une solution dans le réseau simulé, bl_{max} (défini a priori) est le nombre maximum de bypass links pouvant être créés dans le réseau, P est le nombre de partitions restant dans le réseau aprés l'ajout des bypass links et N est le nombre de noeuds dans le réseau global.

Différentes expérimentations ont été réalisées sur des instances de réseaux statiques et dynamiques. Chaque algorithme a été utilisé avec deux types de représentations binaires et deux types de croisement (en deux points et uniforme).

Concernant les instance statiques, nous avons comparé les performances des genGA, ssGA (steady-state GA), CCGA dCCGA (version distribuée du CCGA) et LCGA sur différents réseaux tels que présentés en Figure 22. Pour cela nous avons défini une surface de simulation de 0.2 km^2 et testé trois différents densités de 150, 210 et 350 noeuds par kilomètre carré. Chaque noeud est équipé des technologies Wi-Fi (802.11b) et UMTS. Le rayon de couverture de chaque noeuds est compris entre 20 et 40 mètres pour le Wi-Fi.

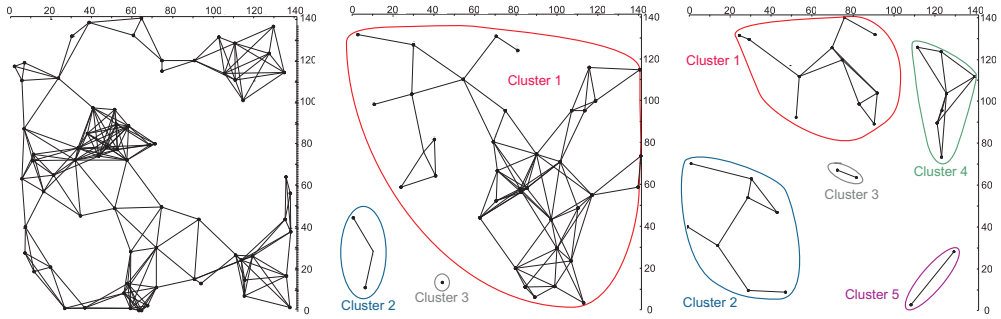


Figure 22: Réseaux étudiés avec 1, 3 et 5 clusters

Ci-dessous nous présentons les résultats obtenus avec le genGA, CCGA et LCGA sur l'instance du problème avec 3 clusters (centre de la Figure 22). Les performances des deux AGCs ont été étudiés avec trois nombres de sous-populations différents (2, 5 et 10). Afin de comparer les performances de ces différents algorithmes, les paramètres présentés dans le Tableau 7 ont été utilisés. Les résultats présentés sont des moyennes obtenues sur 30 expérimentations indépendantes.

Nombre de sous-populations	2, 5, 10 (seulement pour LCGA et CCGA)
Taille des (sous-)populations	100 (genGA, ssGA), 50 (LCGA, CCGA)
Condition de terminaison	50,000 évaluations de fonction
Séléction	Tournoi binaire
Operateur de croisement	Uniforme, p_c=0.8
Operateur de mutation	bit flip, p_m = 1/taille_du_chromosome
Elitisme	1 individu (sauf ssGA)

Table 7: Paramètes utilisés pour le genGA, ssGA, CCGA and LCGA

Il peut être observé graphiquement sur la Figure 23 que les AGCs obtiennent de meilleurs résultats que les deux AG panmictiques, le ssGA procurant les moins bons résultats. En terme de vitesse de convergence, le CCGA a de meilleures performances que les autres AG. Lorsque l'on compare le CCGA et le LCGA, il apparait que le CCGA a de meilleures performances que le LCGA, cette différence accroissant avec le nombre de sous-populations (le meilleur résultat est obtenu par le CCGA avec 10 sous-populations).

Les résultats obtenus sur les instances statiques de ce nouveau problème d'optimisation ont permis de démontrer la supériorité des AGCs sur les AG panmictiques. Différentes analyses sur les performances de algorithmes (résultats, vitesse de convergence, coût de calcul) ainsi que sur le problème

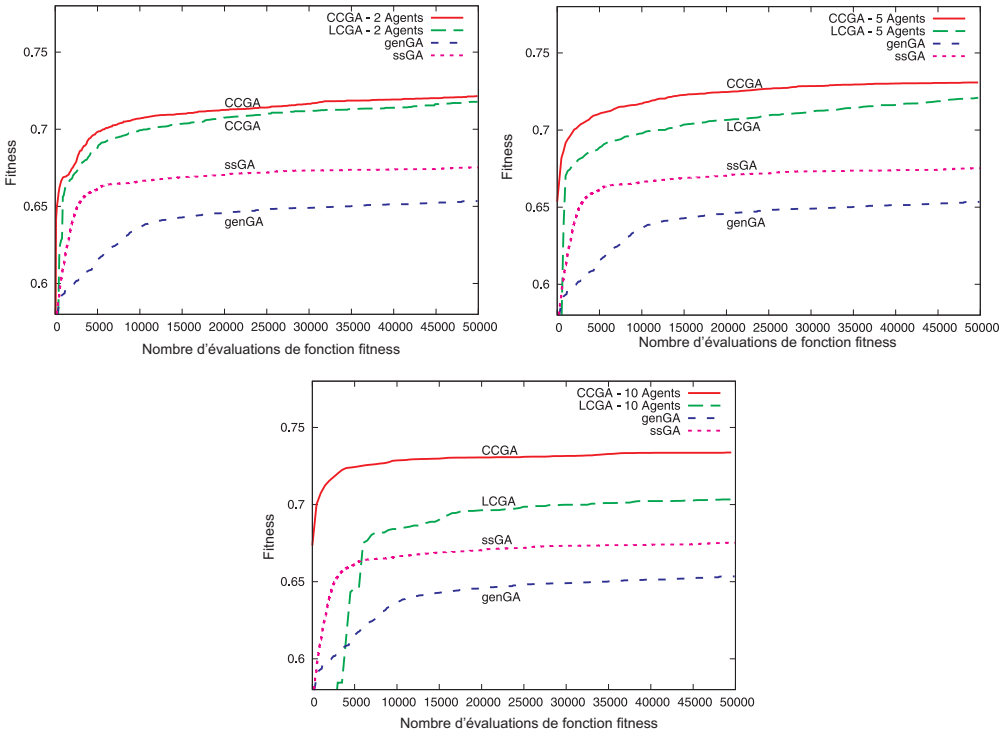


Figure 23: Optimisation du problème des réseaux d'injection avec le genGA, ssGA, CCGA et LCGA

lui-même (nombre de bypass links crées à chaque génération, noeuds le plus souvent élus en tant que points d'injection) ont été réalisées.

De nombreux problèmes métiers sont intrinséquement dynamiques, le problème lui-même ou ses contraintes peuvent changer au cours du temps, l'optimum change alors lui aussi. Dans ce cas, l'objectif de l'algorithme évolutionnaire n'est plus seulement de localiser l'optimum, mais de le suivre au cours du temps. Les réseaux ad hoc étant eux-mêmes intrinséquement mobiles, nous avons donc comparé les performances des genGA, ssGA et CCGA sur une instance dynamique du problème des réseaux d'injection. Chaque algorithme effectue 300.000 évaluations de fonction problème étant modifié toutes les 50.000 évaluations (les noeuds se déplacant slon le modèle de mobilité *random waypoint*). Les algorithmes doivent donc adapter leurs solutions en fonction du problème changeant (c-à-d les 6 différentes états du réseau tels que présentés en Figure 24. La configuration intiale du réseau est identique au réseau statique précédemment étudié (voir centre de la Figure 22). Le modèle de mobilité utilisé est le random waypoint, la vitesse des noeuds étant comprise entre 10 m/s et 50 m/s.

Les résultats présentés à la Figure 25 sont des moyennes obtenues, sur 30 expérimentations indépendantes, par le genGA, le ssGA et le CCGA utilisant la première des deux représentations et les deux types de croisement (2 points et uniforme). Comme pour les instances statiques, il apparait clairement que le CCGA obtient de meilleurs résultats que le genGA et le ssGA tant en vitesse de convergance qu'en meilleurs résultats obtenus (à chaque changement du réseau).

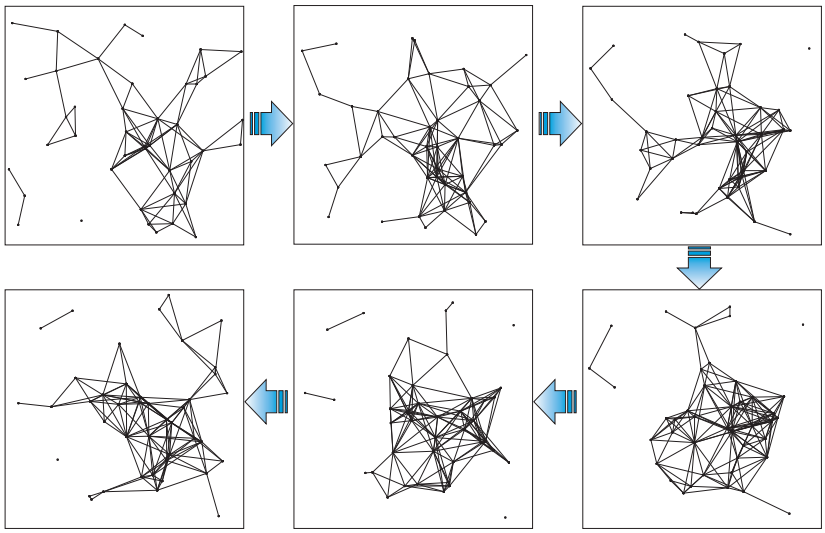


Figure 24: Les six états du réseau d'injection mobile

L'optimisation de ce deuxième problème métier, pour lequel nous avons développé une première modélisation mathématique, a d'une part permis de démontrer experimentallement que les AGCs obtiennent des résultats de référence sur des instances statiques et dynamiques et d'autre part que la plateforme DAFO peut être interfacée avec des logiciels tiers tels que le simulateur de réseaux Madhoc.

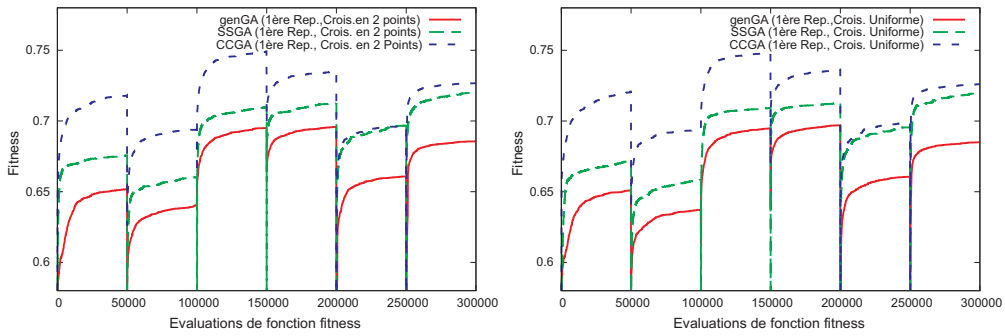


Figure 25: Optimisation du problème des réseaux d'injection dynamiques avec le genGA, le ssGA et le CCGA

Conclusion

Dans cette dissertation nous avons répondu à une partie des limitations inhérentes à l'utilisation des AGCs et plus précisement à leur application sur des problèmes métiers (statiques et dynamiques). Notre objectif à été de construire une plateforme multi-agent permettant l'utilisation, la comparaison et la distribution d'AGCs sur des problèmes d'optimisation.

Dans cette thèse, les contributions suivantes ont été apportées:

- Nous avons démontré que les AGCs ont actuellement été peu utilisés sur des problèmes d'optimisation

métier et que très peu de plateformes actuelles dédiées à l'optimisation évolutionnaire permettent l'utilisation d'AGCs. De plus aucune de ces plateformes n'utilise le paradigme agent pour modéliser, implémenter et distribuer ces algorithmes.
- Nous avons développé MAS4EVO (Multi-Agent System for EVolutionary Optimisation), un nouveau modèle multi-agent dédié à l'optimisation évolutionnaire. MAS4EVO procure une nouvelle façon de modéliser les AGCs en tant qu'organisation d'agents, permettant d'expliciter leurs caractéristiques en terme de topologie, d'interactions, et d'adaptation. Nous avons démontré qu'en utilisant quelques primitives organisationnelles il est possible de définir de multiples organisations et donc de multiples AGCs. En effet, nous avons pu modéliser deux AGC existants (CCGA et LCGA) et deux nouvelles variantes du LCGA (hybride et dynamique).
- La plateforme DAFO (Distributed Agent Framework for Optimization) a été présentée. DAFO est l'implémentation du modèle MAS4EVO qui permet d'appliquer, de comparer et de distribuer les AGCs modélisés avec MAS4EVO avec un minimum de programmation nécessaire. L'utilisation du modèle MAS4EVO permet de faciliter la compréhension et la manipulation des AGCS.
- Deux problèmes métiers ont été optimisés à l'aide de DAFO. Le premier est un problème de gestion de stock existant, pour lequel nous avons experimentallement démontré le gain apporté par la décomposition du problème, le LCGA obtenant de meilleurs résultats que le CCGA sur de petites instances. Nous avons également démontré que les deux variantes du LCGA, dLCGA et hLCGA, procurent de meilleurs résultats que le LCGA "standard". Le second problème métier traité est un nouveau problème de contrôle de topologie dans les réseaux ad hoc hybrides. Nous avons défini un modèle mathématique de ce problème et obtenus des résultats de référence avec les AGCs, que ce soit sur des instances statiques ou dynamiques.

Les perspectives des travaux effectués au cours de cette thèse sont les suivantes:
- L'extension de la plateforme DAFO de part l'ajout de nouveaux algorithmes dans les Problem Solving Agents. Nous nous intéresserons à des algorithmes récents obtenant des résultats de référence sur divers problèmes d'optimisation difficiles, comme les AG cellulaires et les AG multi-objectifs (par ex. Mocell).
- L'appronfondissement de l'étude du problème des réseaux d'injection, en comparant nos AGCs à d'autres AG de références, en optimisant des instances des plus grande taille et en cherchant une décomposition efficace du problème pour le LCGA.
- L'extension du modèle MAS4EVO afin de passer d'une réorganisation du SMA actuellement gérée par l'utilisateur à une réorganisation autonome des agents.

Chapter 1

Introduction

Contents

In this chapter, we first describe the context in which this thesis took place. We then motivate the development of a novel framework for coevolutionary algorithms. We finally present our contributions to the domain and detail the general structure of this dissertation.

1.1 Context

This thesis work takes place in the context of business decisions automation using Evolutionary Algorithms (EAs).

EAs are heuristic methods for solving computationally difficult problems based on the mechanisms of natural selection and genetics. That is, they apply Darwin's principle of evolution (survival of the fittest) among candidate solutions (known as "individuals") with the stochastic processes of gene mutation, recombination, etc. EAs frequently have an advantage over many traditional local search heuristic methods when search spaces are highly modal, discontinuous, or highly constrained and therefore have been applied to a variety of problems, from static optimization to job-shop scheduling.

When applied to business problems optimization, the objectives of EAs can be either to provide a single "very good" solution forever (design engineering, e.g. road network) or to provide "good" solutions in a short time (production optimization, e.g. delivery routes). We are interested in the second problem type which is typically of very high complexity, changing in time (dynamic) and decomposed in multiple subproblems which have to be optimized while satisfying the global corporate goal. Their optimization is therefore decentralized and requires to obtain robust and acceptable

solutions at anytime and in real-time (or at least in a minimum of time). In order to apply efficient EAs on this kind of problem, they have to gain from the problem decomposability, since on the one hand it was demonstrated that considering the problem structure permits a supra-linear acceleration and on the other hand it brings additional information to the problem which can lead to better solutions.

1.2 Motivation

EAs are a class of optimization algorithms which shows key capabilities for business problems optimization. Unfortunately, "classical" EAs tend to perform poorly or are difficult to apply on some problems especially when they have very large search spaces like many real-world problems. These new complex problems are in their nature distributed, i.e. they can be seen as a set of independent interacting entities with their own goals and where a global behavior can be observed as the result of their interactions. In order to address these kinds of problems, researchers referred again to a nature inspired process so as to extend evolutionary algorithms: coevolution (i.e. the coexistence of several species).

The main differences between Coevolutionary algorithms (CEAs) and EAs come from the adaptive nature of fitness evaluation in coevolutionary systems: the fitness of an individual is based on its interaction with other individuals from other so-called subpopulations. Thus, instead of evolving a population of similar individuals representing a global solution like in classical EAs, CEAs consider the coevolution of subpopulations of individuals representing specific parts of the global solution and thus problem decomposition. Intrinsically, by evolving several coadapted subcomponents, CEAs present an opportunity for parallelism (e.g. each subcomponent on a different processor). Such systems have historically been categorized as competitive or cooperative and have nowadays proven to be a popular extension of traditional EAs for test problems (test functions, strategy games, robot games). However, CEAs have still been seldom applied to business optimization problems.

In order to facilitate the use and comparison of EAs, many different libraries and frameworks have been proposed. However, as we will demonstrate in this dissertation, only a few of them allow the use of CEAs. This already partially motivated our research, aiming at providing a new framework dedicated to CEAs. However this was not a sufficient motivation to start building a new framework from scratch, since extending one of the available frameworks would have been sufficient.

The second reason lies in the model used in the existing frameworks which is in the vast majority object-oriented whereas in the literature describing CEAs most often subpopulations are described as agents. Therefore our second motivation dwells in taking benefit from the multi-agent domain and to model our framework (and thus the CEAs) as a multi-agent system (MAS). It consequently makes explicit: the structure of the CEAs (e.g. the interactions between agents), the dynamics inside this

structure and the interactions with the system's environment (i.e. with the optimization problem). Additionally, it is possible to profit from existing multi-agent platforms so as to ease the development and the distribution of our framework.

1.3 Contributions

The main contributions of this dissertation are as follows:

- MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, a new agent organizational and reorganizational model based on Moise+ and dedicated to evolutionary optimization. It provides a novel way of modeling and implementing CEAs.
- DAFO, Distributed Agent Framework for Optimization, being the implementation of the MAS4EVO model. Built on top of a multi-agent platform, it allows us, with few coding efforts, to use, distribute and compare various CEAs (existing and novel ones) on optimization problems such as business problems.
- The creation of two new variants of a competitive CEA, a hybrid and a dynamic one and their integration into our DAFO framework. The study and the comparison of their performance to "standard" CEAs on the business-problems described in the following point.
- The application of CEAs (existing and new ones) and their comparison on two business problems. The first problem tackled is a stock management for which we studied multiple static instances. The second problem studied is a new topology control problem in wireless ad hoc networks. The performance of different CEAs was studied on multiple static instances and on one dynamic instance of this network optimization problem.

1.4 Dissertation Outline

This dissertation is organized in three main parts, the first one (chapters 2 and 3) provides a state-of-the-art respectively on CEAs and on organizational models in MAS. The second part (chapters 4, 5 and 6) describes in detail the DAFO framework, i.e. its new agent organizational model, the new CEAs it features and its implementation. Finally the third part (chapters 7 and 8) presents two business optimization problems tackled using DAFO and the CEAs it provides.

Chapter 2 provides a brief overview of EAs followed by an description of parallel and multi-populations variants before introducing the different available CEAs. The latter are studied in detail as well as their different fields of applications. An investigation follows on the usage of CEAs on business problems and on the existence of hybrid and/or dynamic versions in the literature. Finally EAs platforms are analyzed in terms of model (object or agent oriented) and of proposed algorithms

(i.e. the availability of CEAs).

Chapter 3, after a short introduction to the agent paradigm, provides a survey on multi-agent organizations and organizations' adaptation in multi-agent systems. This chapter provides the necessary material to choose a model which will explicit the structure of the CEAs, their dynamics and their interaction with the environment (i.e. with the optimization problem which itself can be dynamic).

Chapter 4 describes MAS4EVO, a new multi-agent model dedicated to evolutionary optimization. Its agent, interaction and environment models are presented as well as its organizational model extending Moise+. Two state-of-the art CEAs available in DAFO are then modeled using MAS4EVO.

Chapter 5 provides a description of two new competitive CEAs, a hybrid one and a dynamic one, and their model using MAS4EVO.

Chapter 6 gives details concerning DAFO, the framework implementing the MAS4EVO model. Its architecture, its distribution and its dedicated description language are presented.

Chapter 7 introduces the first business optimization problem tackled using DAFO, which is a stock management problem called ICP (Inventory Control Parameter) problem. Some experimental results obtained comparing a competitive and a cooperative CEA are presented. Next, the performances of the new hybrid and dynamic competitive CEAs are analyzed and compared to the "standard" CEAs on the ICP problem.

Chapter 8 describes the second business problem optimized using DAFO: a topology control problem in hybrid wireless ad hoc networks called the injection network problem. Key related works are presented before giving a detailed view on the injection network problem for which a fitness function was defined. Experiments using CGAs including a distributed version are conducted on several static instances and one dynamic instance of the problem and results are discussed.

The last chapter presents our conclusions and the perspectives this thesis work offers.

Part I

State of the art

Chapter 2

Coevolutionary Genetic Algorithms (CGAs)

Contents

Since the 1940s/1950s and the emergence of modern computers, the idea of mimic some of the nature's mechanisms to create Artificial Intelligence (AI) has captured the imagination of many computer scientists. Many fields of research have arisen in the pursuit of these ideas. One of these fields, as found in the computer science and engineering domains, is termed "evolutionary computation" (EC), the use of self-evolving strategies in problem solving. Among the tools available in EC, one is the genetic algorithm (GA).

Genetic algorithms were first introduced by John H. Holland in the mid 1970s [4]. Holland and some of his students (e.g. K. DeJong) examined the capacity of biological systems to change and adapt at the genetic level in response to environmental problems and challenges. As a result, they introduced the basic concepts behind the theories of genetic algorithms, which are an attempt to apply similar mechanisms occurring in nature to scientific problems.

Since then, GAs have been extensively used for many optimization problems, from test functions to complex business problems. Since the 1990's, a new type of GAs called Coevolutionary GA (CGA) has emerged and has become a very active research area.

One aspect of our research is focused on using some existing CGAs and building new hybrid and dynamic variants in order to optimize real-world problems. For this reason, in the coming chapter (2.2) we start by providing an introduction to genetic algorithms followed by an overview of parallel and multi-populations variants in 2.1.2. Afterwards, in 2.2 we study in detail the different CGAs available, their different fields of applications. We further investigate if some CGAs were used on business problems and if hybrid and/or dynamic versions have been proposed in the literature.

A second aspect of our research consists in building a framework dedicated to CGAs. The latter has to be easily utilizable, tunable and must provide adaptation capabilities to the user and/or to the system itself. To this aim, in 2.3 we examine existing Evolutionary Algorithms (EAs) platforms so as to analyze if they allow the use of such CGAs and if they use the agent paradigm.

2.1 Genetic Algorithms (GAs)

Different approaches exist in optimization and researches. As presented in Figure 2.1, we can put those algorithms into three classes: calculus based approaches (Greedy, Fibonacci), enumerative techniques (Branch&Bound, Primal Simplex) and random approaches (Tabu Search, Neural Networks, Evolutionary Algorithms). In the case of business optimization problems, the complexity is such that enumerative techniques from the classical operational research are not adapted. Calculus based algorithms (complexity of O(n)) provide rapid solutions but too far from the global optimum. Then it just remains the class of random based algorithms. According to the quality of the solution, most of

the algorithms are valuable as depicted in the famous "no free lunch theorem" [3].

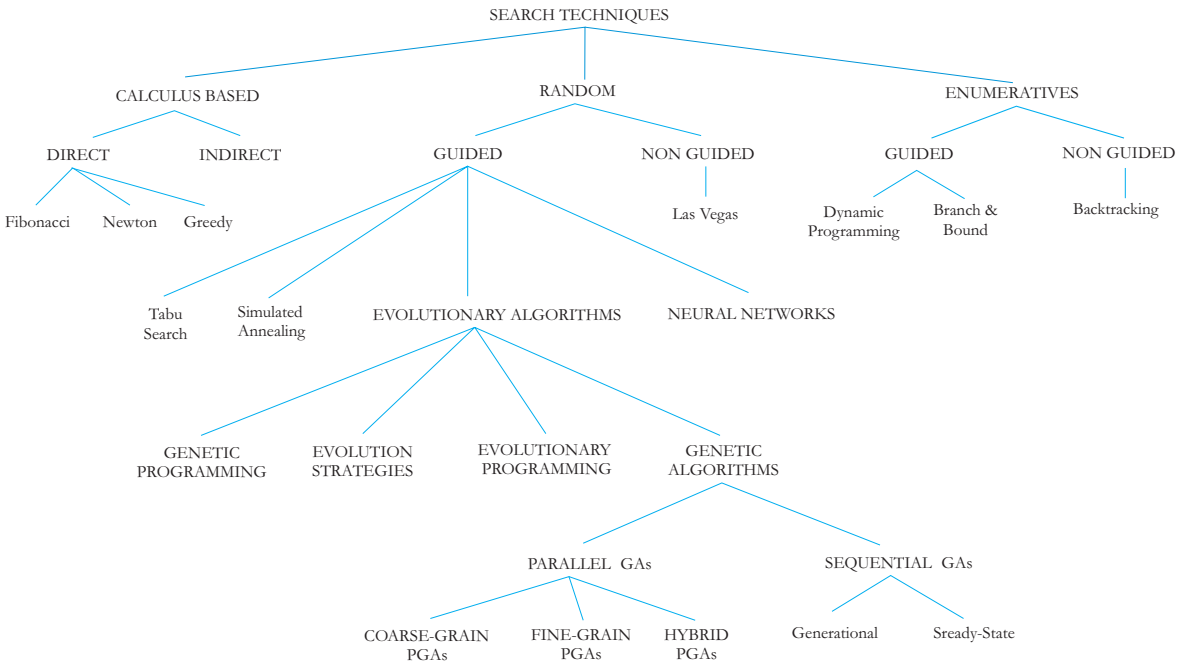


Figure 2.1: Taxonomy of Search Techniques from [1]

Genetic Algorithms (GAs) are part of the Evolutionary Algorithms (EAs) which are optimization and learning techniques based on the Darwinian evolutionary model. EAs are powerful heuristic methods for solving many types of computationally difficult problems. The fascinating nature and often surprising successes of EAs have drawn researchers to the field for the better part of half a century. Additionally to GAs, EAs group Evolution Strategies (ES), Genetic Programming (GP)and Evolutionary Programming (EP). These adaptive approaches are common in that they derive their inspiration from the natural processes whereby biological organisms evolve. Upon a little reflection it seems a reasonable suggestion that biological evolution, as set out by the Darwinian theory of natural selection, is a powerful adaptive process by which organisms dynamically improve their suitability to the surrounding environment.

2.1.1 Sequential Genetic Algorithms

Sequential genetic algorithms are chronologically the first GAs and also the simpler ones. They operate in an iterative manner, generating new populations of individuals from the old ones. In this section we will describe two sequential GAs, the generational GA and the steady-state GA, the representation issue and some of the genetic operators they can use.

2.1.1.1 Generational GA

The use of evolutionary computation (EC) techniques to evolve solutions for both abstractions and real-life problems has seen a dramatic increase in popularity and success over the last decade. The most popular and widely applied EC technique is the sequential GA [33], whose computational scheme is based on a set (population) of potential solutions (individuals) on which it applies some stochastic operators in order to search for an optimum. It uses a single population (panmixia) of individuals and apply operators to them as a whole.

Algorithme 2 : Generational GA

```
Generate initial population P_t
Evaluate population P_t
while Stopping criteria not satisfied do
    for i = 1 to pop_size do
        Select individuals i_1 and i_2 from P_t
        i' = Crossover(i_1, i_2)
        i" = Mutate(i')
        Insert individual i" in P_{t+1}
    end
    P_t = P_{t+1}
    Evaluate population P_t
end
```

The generational genetic algorithm (also referred as "Standard Genetic algorithm") [5], see Algorithm 2 and Figure 2.2, begins by initializing a population of individuals (genotypes). A random initialization is commonly used, however, for some applications knowledge may be available to enable the population to be more intelligently initialized. Each genotype is then decoded into a problem solution instantiation (phenotype) and its fitness evaluated. Individuals are then selected non-deterministically, based on their fitness, to reproduce. Once a reproductive pool has been selected, recombination (also called crossover) is applied to create offspring and the offspring are mutated. Next, the fitness of each offspring is evaluated. Finally, old population members are, with equal likelihood, randomly replaced with the offspring to produce a new population. This select, recombine, evaluate, and replace cycle continues until a predefined termination condition (e.g. a number of fitness function evaluations). As the algorithm runs, selection allows the algorithm to focus on regions of the solution space with an above average observed fitness, this phenomenon is called exploitation. Genetic operators (crossover and mutation) enable the algorithm to explore new regions of the solution space, it is the exploration. The combination of exploitation and exploration makes the algorithm evolve and converge to better solutions. Past works have shown that the underlying iterative step of the GA is very influent in some applications [34].

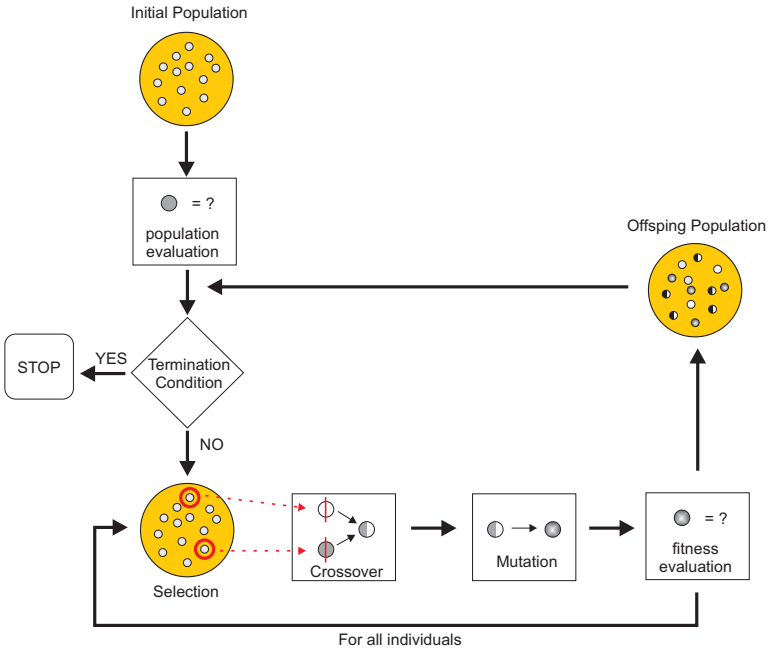


Figure 2.2: GA Functioning

2.1.1.2 Steady-State GA

The incremental/steady state genetic algorithm (ssGA) [6] differs from the generational model in the sense that only a few individuals are replaced in each generation (one single individual in our case). A replacement/deletion strategy defines which member(s) of the population will be replaced by the new offspring(s). This strategy can be to replace the least fit individual or the oldest one in the population by the offspring resulting from crossover and mutation of the selected individuals (see Algorithm 3).

Algorithme 3 : Steady State GA

```
Generate initial population P
Evaluate population P
while Stopping criteria not satisfied do
    Select individuals i_1 and i_2 from P
    i' = Crossover(i_1, i_2)
    i" = Mutate(i')
    Evaluate (i")
    Insert individual i" in P
end
```

2.1.1.3 Representation

Before a genetic algorithm can be used to solve a problem it is necessary to define a representation and a mapping between the representation and problem solutions. In biology, the representation (i.e. the genetic code) of an organism is referred to as its genotype, and the instantiation of this code,

that is, the physical realization of the being, is referred to as the organisms phenotype. The choice of the representation is an important step in the genetic algorithm development process since it will influence the performance of the evolutionary process (it can affect the speed of convergence and even the quality of the solutions evolved). The chromosomes are normally strings, but they could even be multidimensional arrays, or trees. They can be of fixed or variable lengths. A binary notation is often used but GA's are not restricted to binary representations.

2.1.1.4 Genetic Operators

As described in the two previous sections (2.1.1.1 and 2.1.1.2), the evolutionary process of GAs includes three genetic operators, namely selection, crossover and mutation. This section provides a brief description of these operators and of their different variants.

Selection

During the selection process, the individuals producing offspring are chosen. Each individual in the selection pool receives a reproduction probability depending on the own objective value and the objective value of all other individuals in the selection pool. This fitness is used for the actual selection step afterwards. There are many methods in selecting the chromosomes, here is a brief description of some of them.

- Roulette wheel selection: simplest selection scheme, also called stochastic sampling with replacement. The chance of a chromosome to get selected is proportional to its fitness (or rank). This is where the concept of survival of the fittest comes into play.
1. Sum the total fitness values of the individuals in the population, where sum = T
2. Repeat N times for population P of size N
3. Generate a random integer r, with $0 < r < T$
4. Step trough the population and sum the fitness values until the current sum is >=r. The individual whose value shifted the sum over the limit is the one selected.

- Rank selection: rank based selection was designed to overcome the disadvantages of the roulette-wheel method. In this scheme, individuals are sorted according to their fitness value and a rank N-1 is assigned to the best individual and a rank of 1 to the least fit individual. A selection probability is then linearly assigned to each individual according to the rank value.

- Tournament selection: a tournament is repeatedly held in which N individuals are selected for the current population (often N=2) and the fittest individual is copied into the intermediate population. The process is repeated until a new population is created. A variant called soft tournament exists, in which the winner is accepted with some predefined probability.

Crossover

The crossover operation, also named as recombination, is a genetic operator that combines (mates) two chromosomes (parents) with a fitness dependant probability (typically between 0.6 and 1), to produce a new chromosome (offspring).

- One-point crossover: a crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring. Figure 2.3 shows an example of a one-point crossover recombining two binary chromosomes, the crossover point being represented by the red line.

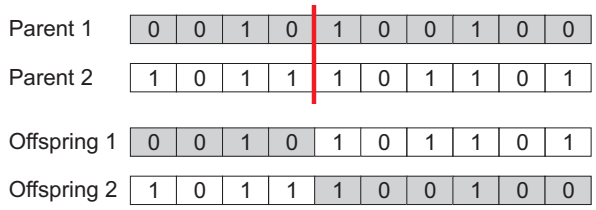


Figure 2.3: One Point Crossover

- Two-point crossover: in two-point crossover, two crossover positions are selected uniformly at random and the values between these points are exchanged. Then two new offspring are produced. Figure 2.4 shows an example of a two-point crossover recombining two binary chromosomes, the two crossover points being represented by the two red lines.

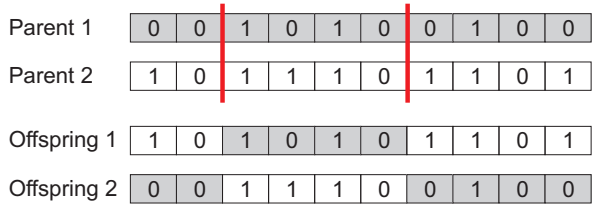


Figure 2.4: Two Point Crossover

- Uniform crossover: single and multi-point crossover define cross points as places between loci where an individual can be split. Uniform crossover [35] however generalizes this scheme to make every locus a potential crossover point. A crossover mask, the same length as the individual structure, is created at random and the parity of the bits in the mask indicates which parent will supply the offspring at every position.

Consider the following two individuals presented in Figure 2.5 with 10 binary variables each. For each variable the parent who contributes its value to the offspring is chosen randomly with equal probability. Here, the offspring 1 is produced by taking the bit from parent 1 if the corresponding

mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0. Offspring 2 is created using the inverse of the mask, usually.

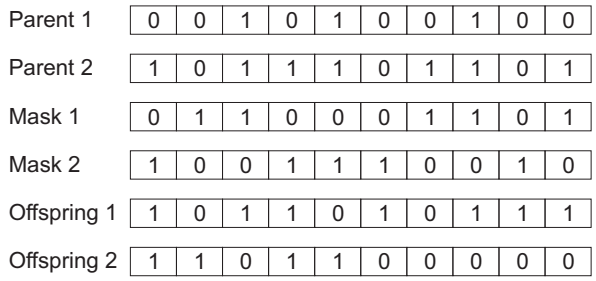


Figure 2.5: Uniform Crossover

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation, and the particular coding for a given parameter set.

Mutation

The mutation operator is introduced to prevent premature convergence to local optima by introducing some randomness into the search. If a binary encoding is used, mutation is carried out by changing a 1 into a 0 and vice-versa, which is known as bit-flip mutation (as shown in Figure 2.6). Most of GAs apply mutation with a constant low probability (typically between 0.001 and 0.01).

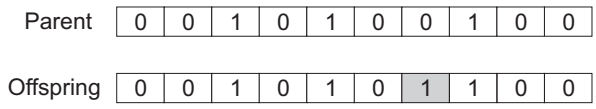


Figure 2.6: Bit-flip mutation

2.1.2 Parallel Genetic Algorithms

PGAs are parallel stochastic algorithms based on natural evolution theory. Better individuals survive and reproduce themselves more often than the worse ones. To speed up the processing of generations of populations, it is possible to use a code parallelizer embedded in the compiler (for more information see [1]), or else by the explicit parallelization (master-slave global parallelization) of the genetic operators and/or evaluations.

Adding parallelism to GAs implies interesting features as described in [36]

- the reduction of the time to locate a solution (faster algorithms)
- the reduction of the number of function evaluations (cost of the search)
- the possibility of handling larger populations using parallel platforms for running the algorithms

- the improved quality of the solutions worked out.

Alba in [1] has shown that PGAs are also less prone to premature convergence to sub-optimal solutions.

PGAs are not just parallel versions of sequential GAs. It has been proven that a PGA behaves better than the sum of the separate behaviors of its component sub-algorithms (super-linear speedup) [37]. PGAs then represents a new algorithmic class providing a different and often better search mechanism, i.e. PGAs are a class of guided random evolutionary algorithms, as can be seen in the taxonomy presented in Figure 2.1.

A lot of different models of PGAs have been proposed and many classification, surveys and overviews have been published. Alba and Troya in [1] and Alba and Tomassini in [7] proposed a survey of parallel models and implementations of EAs. Those papers stress algorithmic issues but also tools for building PEAs. They provide a good theoretical background and many useful links. Konfrst in [8] gives a good overview of theoretical advances (most significant works before and after 2000), of trends in computing (architectures, OS ad topologies, libraries and programming), applications (most significant works since 2000).

A common classification has been found where all those PGA models fit into more general classes, where PGAs are divided into global, coarse-grained, fine-grained and hybrid models as presented in Figure 2.7.

- Single population master-slave model (Figure 2.7 (a)): offers the easiest and simplest way of parallelization of single population GAs. A Master-processor runs GA performing selection and genetic operators. The operation of fitness evaluation is parallelized on slave-processors. This model requires global synchronization and control of the GA.

- Coarse-grained (Figure 2.7 (b)): Coarse-grained GAs are known with multiple names as mentioned by Cantú-Paz in [38]. They are also called multiple demes GA (where a deme refers to a sub-population), and island model. They consist on several subpopulations, each subpopulation running similar GAs, that exchange individuals periodically with other subpopulation. This exchange of individuals is called migration and is controlled by several parameters (intervals, size, etc.). The structure of the population is defined by the topology of a communication graph, which specifies a neighborhood of each subpopulation. The island model strategy eliminates the global synchronization that is required in the panmictic approach. However, some synchronization is usually required for migration.

- Fine-grained parallel GA (Figure 2.7 (c)): Fine-grained parallel GAs (also called cellular model or diffusion model) have only one population, but it is has a spatial structure that limits the interactions between individuals. An individual can only compete and mate with its neighbors, but while the island model has fixed boundaries between its subpopulations and a structured

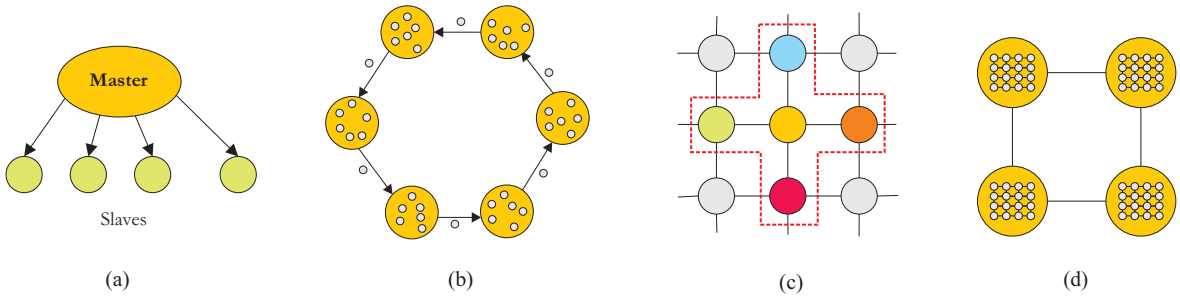


Figure 2.7: Different PGA Models: (a) master-slave, (b) coarse grain, (c) fine grain, (d) hybrid (coarse grain and fine grain)

mechanism for migration, the cellular model has overlapping demes and incorporates migration to the same effect without the overhead involved (i.e. good solutions may disseminate across the entire population). Different local mating strategies and different neighborhood sizes and shapes can be used. It achieves the same effect of isolation found in the island model by using isolation by distance. In this case, new species are formed at the boundaries between emergent structures. The cellular model also avoids the problems associated with the panmictic approach, such as global synchronization. However, cellular models require a large amount of communication.

- Hierarchical (Hybrid) Parallel GA (Figure 2.7 (d)): combine on two levels different PGAs, producing hierarchical parallel GAs. Some of these new hybrid algorithms add a new degree of complexity to the already complicated scene of parallel GAs, but other hybrids manage to keep the same complexity as one of their components. When two methods of parallelizing GAs are combined they form a hierarchy. At the upper level most of the hybrid parallel GAs are multiple-population algorithms.

2.1.2.1 Fine-Grain PGA

Cellular GAs usually assume a synchronous (or "parallel") update policy, in which all the individuals are formally updated at the same time. However, the alternative is to use an asynchronous (or "sequential") update method, in which the offsprings are directly placed in the current population by following some rules instead of updating all the individuals simultaneously. The shape of the structure in which individuals evolve has also a deep impact on the performance of the cGA.

Alba and Troya in [39] have compared the properties of steady-state, generational and cellular GAs. The analyzed features where time complexity, selection pressure in centralized and decentralized models, schema processing rates in panmictic and structured GAs, efficacy in finding the optimum, efficiency, speedup and resistance to scalability. In [40], the reader can find a study of the influence of some different asynchronous policies and grid shapes on the problem solving capabilities of cGAs. In [41], 19 different cGAs are tested on the well-known benchmark SAT problem. Asynchronous

adaptive algorithms are in this case the most effective. Dynamic grid changes appears to favor the exploration/exploitation capabilities of the algorithm and thus helps it to avoid getting stuck in local optima in many cases. Future works in this field will focus on trying other criteria for the adaptation of the shape of the population.

2.1.2.2 Coarse Grain PGA

When dealing with multiple demes, several new parameters have to be taken into account concerning topology and migration.

Migration: In most of the multiple demes GA, the migration process is synchronous which means that migration occurs at predetermined constant intervals. Migration can also be asynchronous, like in Grosso's dissertation [42] where a delayed migration scheme is introduced in which migration is enabled until the population is close to converge.
In each case it is necessary to determine when the migration will occur and how migrants are selected/incorporated from/to the source/target islands.

Topology: The topology is an important factor in the performance of the parallel GA because it determines how fast (or how slow) a good solution disseminates to other demes. If the topology has a dense connectivity (or a short diameter, or both) good solutions will spread fast to all the demes and may quickly take over the population. On the other hand, if the topology is sparsely connected (or has a long diameter), solutions will spread slower and the demes will be more isolated from each other, permitting the appearance of different solutions. These solutions may come together at a later time and recombine to form potentially better individuals. The general trend is to have a static topology specified at the beginning of the run and that is not changed. Another trend is to use dynamic topologies, where a deme is not restricted to communicate with a fixed neighborhood but migrants are sent to demes that meet some specific criteria (i.e. measure of diversity, measure of genotypic distance between the two populations, etc.).

Cantú-Paz has brought new principles in multiple dissertations the most famous being maybe [38]. It helped many researchers to choose a configuration (topologies, migration rates, number and size of demes) for their PGA. In [43] Cantú-Paz has compared different migration policies (best migrants replace worst individuals, best migrants replace random individuals, random migrants replace worst individuals) and he showed that the choice of migrants and the replacement of individuals are important parameters that give significant improvements when chosen according to their fitness. In [44] he showed the relation between the deme size, the migration rate and the degree of the topology with the probability of success after two and multiple epochs. According to him, the topology has also a significant impact on the solution quality but different topologies with the same degree reach almost identical solutions. He also proposed a quality model that can be used as a deme sizing

equation, which in turn can be used to find the degree and number of epochs that minimize the execution time. In [45], he additionally brought to light that isolated demes should be avoided in practice.

There are still many ongoing works on this subject, like in [46] where De Jong and Skolicki investigate the influence of migration sizes and intervals on island models. They have shown that the island model is very sensitive to small migration interval and that a migration size of 10 percent is unnecessarily big. In their case, the best performance is achieved with moderate migration intervals and small migration sizes. Too frequent migrations should be avoided, because all islands start to share the same individuals and results in a loss of diversity that negatively affects performance. One of their current hypothesis is that migrations should occur only after the diversity in the system stabilizes.

2.1.2.3 Hierarchical PGA

In hierarchical (or hybrid) parallel GAs, two levels of parallelizing GAs are combined. At the upper-level, most of the hybrid parallel GAs are multiple-population algorithms (coarse-grain implementation). Some hybrids have a fine-grained GA (i.e. cGA) at the lower level, as shown in figure 2.1.2.(d).

This model was first introduced by Gruau in [47] and applied on a neural network design and training application. Many works followed based on this hierarchical model like in [48] [49] or more recently in [50] and [51]. Another type of hierarchical parallel GA uses a master-slave on each of the demes of a multi-population GA. This approach is useful for complex problems requiring high computational power and was for instance used by Goldberg in [52] and later by Bianchini in [53]. The last hierarchical model is composed of coarse-grained at both levels. A high migration rate and a high density is used inside the demes so as to force panmicitic mixing while a low migration rate is used at the upper level. For more details the reader can refer to [7] and [38].

2.1.2.4 Synthesis

PGAs have been developed in order to overcome the drawbacks of sequential GAs when tackling non-trivial problems of big sizes (e.g. big computational time, big memory usage, etc.). An interesting feature of PGAs is that they are not only parallel versions of a sequential algorithm meant to speed up the computational process. Indeed, they are a new class of meta-heuristics which structured population and parallel execution provides a higher efficiency and efficacy.

Those features led to an high number of applications of these algorithms (applications that we did not listed here, the reader can refer to [1] and [8]) as well as the development of many frameworks, some of the most prominent ones being studied in section 2.3.

However, as for sequential GAs, PGAs still consider the evolution of similar individuals representing the global solution. They do not allow decomposition, neither at the level of the solution representation nor at the level of the problem itself, while in some cases this decomposition can provide interesting

improvements in terms of solution quality and speedup. We therefore studied another class of GAs which takes into account these characteristics, i.e. coevolutionary GAs.

2.2 Coevolutionary Genetic Algorithms

A very natural, and increasingly popular extension when problems domains are potentially complex, or when it is difficult or impossible to assess an objective fitness measure for the problem, is the class of so-called coevolutionary algorithms (CEAs) [10]. This ostensibly allows the potential for evolving greater complexity by allowing pieces of a problem to evolve in tandem, as well as the potential for evolving solutions to problems in which such a subjective fitness may, in fact, be necessary (i.e., game playing strategies). In a general sense, coevolution refers to a reciprocal evolutionary change between species that interact with each other. The term "coevolution" is usually attributed to Ehrlich and Raven who published a paper on their studies performed with butterflies and plants in the mid-1960s [54]. However, several other researchers used the term before them. In fact, the original idea of co-evolution was introduced by Darwin [55].

Instead of evolving a population of similar individuals representing a global solution like in classical GAs, CGAs consider the coevolution of subpopulations of individuals representing specific parts of the global solution. Researchers have developed several coevolutionary approaches which normally involve two or more species, in most cases each species independently runs a genetic algorithm. Individuals are evaluated based on their direct interactions with other individuals, these interactions can be either positive or negative depending on the consequences that such interaction produces on the population. If negative, the presence of each species is associated with reducing the growth of another species, this is competitive coevolution. If positive, the presence of each species stimulates the growth of the other species, this is cooperative coevolution. Consequently, there are two main classes of coevolutionary algorithms in the evolutionary computation literature:

- The competitive CGAs: the fitness of an individual is the result of a series of encounters with other individuals.
- The cooperative CGAs: the fitness of an individual is the result of a collaboration with individuals of other species (or populations).

Table 2.1 summarizes the differences between the two multi-population approaches that are the island model previously introduced in 2.1.2.2 and CGAs. The island model doesn't support the type of interaction between subcomponents required for them to coadapt and form competitive, exploitative or cooperative relationships. Although the island model improves the performance of EA by maintaining some diversity in the ecosystem and providing more explicit parallelism, it does not address any of the other issues related to the evolution of coadapted subcomponents.

The following two sections, 2.2.1 and 2.2.2, provide a description of those two architectures of CGAs, respectively competitive and cooperative, and of their applications.

Island Model	CGAs
Static number of populations	No restriction on the number of subpopulation (allow dynamic creation or destruction of subpopulations)
Only migration of individuals : mixing of genetic material (reinject diversity into otherwise converging subpopulations)	Individuals do not migrate: interbreeding does not occur.
Predetermined migration interval + migration size	Communication between subpopulations is limited to occasional communication of representatives or other information
Eliminates global synchronization but still synchronization for migration	The only global control is that required to create or eliminate unproductive subpopulations
Well adapted for linearly separable problems	Coevolution is applicable to a wide range of decomposable problems

Table 2.1: Comparison between Island Model and CGAs

2.2.1 Competitive Architecture and Applications

This section presents a (non-exhaustive) list of different competitive architectures and their applications on test and business problems. The choice of not considering only real-world applications is motivated by historical factors (the first architectures were applied on test problems) and by numerical factors (only few applications exist on business problems).

Competitive CEAs were first introduced by Hillis [56] in 1991. Hillis used a model of host and parasites, also called predator-prey, coevolution for evolving sorting networks. One species (the hosts) represents sorting networks, and the other species (the parasites) represent test cases in the form of sequences of numbers to be sorted. An individual in the hosts population (i.e. a sorting network) is awarded a fitness score based on how well it sorts an opponent data set from the parasites population (i.e. a test case), and an individual in the second population represents potential data sets whose fitness is based on how well they confuse opponent sorting networks.

The term Coevolutionary Genetic Algorithms (CGAs) was introduced in 1994 by Paredis in [57] and [58], in which a competitive approach was used respectively for a well known Constraint Satisfaction Problem (CSP), the N-queens problem, and on a classification neural network problem. Other machine learning problems were tackled using competitive approaches like intertwined spirals in [59] or neural networks in [60].

A two-species model has also been used to solve a number of game learning problems, including tic-tac-toe, nim, backgammon and go, by having the species represent competing players [61] [62] [15]. In those models, the two species represent opponents in the game. Each member of one species is matched against a sample of opponents from the previous generation of the other species. An indi-

vidual is evaluated on the number of opponents from the sample it defeats. Finally, the amount of reward resulting from each win is a function of the number of other individuals in the population who can defeat the same opponent.

Additionally Angeline and Pollack [16] demonstrated the effectiveness of competition for evolving better solutions in learning of backgammon strategy by developing a concept of competitive fitness to provide a more robust training environment than independent fitness functions.

Competition was also successfully harnessed by Schlierkamp-Voosen and Mühlenbein [63] to facilitate strategy adaptation in their so-called breeder genetic algorithms which they used for optimizing classical test functions (i.e. De Jong's test suite). Competition has also played an important role in the field of coevolution of complex agent/robots behaviors [64] [65] [66].

Finally, Seredynski in [12] introduced another competitive model called LCGA (Loosely Coupled Genetic Algorithm) motivated by a non-cooperative model of game theory, a variant of the N-person prisoner dilemma game called game with limited interaction. His algorithm was applied to dynamic mapping problem and scheduling problem [13] to a distributed scheduling problem [67] and to test functions [68]. More details on LCGA are provided in section 2.2.5.

Table 2.2 presented hereafter chronologically lists the applications of competitive coevolutionary architectures on both test and real-world problems.

Author	Date	Application
Hillis	1991	Sorting networks
Angeline	1993	Game learning problem (Backgammon strategy)
Paredis	1994	CSP, N-queens problem, classification neural network problem
Schlierkamp-Voosen	1994	Test functions optimization (De Jong's test suite)
Sims	1994	3D morphology and behavior evolution
Rosin	1995	Game learning problem (tic-tac-toe, nim and go)
Juille	1996	Machine learning problem (intertwined spirals)
Rosin	1997	Game learning problem (3D tic-tac-toe and nim)
Seredynski	1997	Dynamic mapping problem and scheduling problem
Luke	1997	Soccer softbots team coordination
Floreano	1997	Coevolutionary Predator-Prey Robots
Mayer	1998	Machine learning problem (neural networks)
Pollack	1998	Game learning problem (backgammon strategy)
Seredynski	1999	Distributed scheduling problem
Seredynski	2003	Test functions optimization

Table 2.2: Competitive Architectures Applications

These competitive-species models have demonstrated that this form of interaction helps to preserve genetic diversity and results in better final solutions when compared with non-coevolutionary approaches. However, one limitation of these approaches is the requirement of a hand-decomposition

of the problem into two antagonistic subcomponents (for the two species model) or into loosely coupled subproblems (for the LCGA).

2.2.2 Cooperative Architecture and Applications

This section presents a (non-exhaustive) list of different cooperative architectures and their applications on test and business problems. As for the competitive architectures, considering not only real-world applications is driven by historical factors (the first architectures were applied on test problems) and by numerical factors (only few applications exist on business problems).

Other researchers have explored the use of cooperative-species models. Husband and Mills in [69] first used cooperative coevolution for a job-shop scheduling problem. The decomposition used is to have all species but one evolving plans for manufacturing a different component. The single remaining species evolves an arbitrator for resolving conflicts when two or more plans required the same piece of shop equipment at the same time.

Cooperative coevolution gained popularity with Potter and De Jong who proposed a general framework for such cooperative models in [11] and applied it to test functions optimization problems (De Jong's test suite). In this coevolutionary approach, multiple instances of GAs are run in parallel, each population contains individuals representing a component of a larger solution. Complete solutions are obtained by assembling representatives from each of the species. Unlike the island model, the individuals from the separate subpopulations do not interbreed. Credit assignment at the species level is defined in terms of the fitness of the complete solutions in which the species members participate. This provides evolutionary pressure for species to cooperate rather than compete. However, competition still exists among individuals within the same subpopulation.

Cooperative coevolution showed to be efficient on different problems like static function optimization [70], rule learning [71] [72], neural network learning [73], and multiagent learning problems [17].

In 1995, Paredis applied a two-species cooperative model on Goldbergs three-bit deceptive function in [74]. This two-species system assigns one species to the task of evolving an effective representation in the form of a mapping between genes and subproblems, and the other species to the task of evolving subproblem solutions. These two species had a symbiotic relationship in that the second species used the representations coevolved by the first species.

Moriarty and Miikkulainen in [75], in the SANE (Symbiotic, Adaptive, Neuro-Evolution) system, took a different, somewhat more adaptive approach to cooperative coevolution of neural networks. In this case a parent population represents potential network plans, while an offspring population is used to acquire node information. Plans are evaluated based on how well they solve a problem with

their collaborating nodes, and the nodes receive a share of this fitness. Thus a node is rewarded for participating more with successful plans, and thus receives fitness only indirectly.

Potter's methods have also been used by Eriksson and Olsson [14] who have used a CCGA for inventory control parameter optimization (i.e. a stock management problem).

Wiegand introduced DCCGA (Diffusable CCGA) in [76] attempts to make the algorithm more adaptively allocate resources by allowing migrations of individuals from one population to another in a method similar to the Schlierkamp-Voosen and Mühlenbein [63] competitive mechanisms. He applied his DCCGA on the same test functions used by Potter in [11].

Table 2.3 presented hereafter chronologically lists the applications of cooperative coevolutionary architectures on both test and real-world problems.

Author	Date	Application
Husband and Mills	1991	Job-shop scheduling problem
Potter	1994	Test functions optimization
Paredis	1995	Goldbergs three-bit deceptive function
Potter	1995-97	Rule learning
Potter	1997	Static function optimization
Eriksson and Olsson	1997	Inventory control parameter optimization
Moriarty and Miikkulainen	1997	Neural networks
Wiegand	1998	Test functions optimization
Potter	2000	Neural network learning
Potter	2001	Multiagent learning problems

Table 2.3: Cooperative Architectures Applications

2.2.3 Synthesis

Purely cooperative CEAs can behave quite differently than purely competitive ones, exhibiting different advantages and disadvantages. However, once a particular algorithm and problem domain are dissected for analysis purposes, it becomes clear that there are often elements of both cooperation and competition in many CEAs. Indeed, when one considers single-population CEAs, it is difficult to discern the difference between competition as a result of selection within the population, and competition as a result of the relationships in the subjective fitness assessment. To conclude, the differentiation and thus the categorization of CEAs are not always trivial.

Considering the application of CGAs, both competitive and cooperative, we can notice that most of them are targeted to test problems. Up to now, few works tried to benefit from CGAs on real-world problems.

The last two sections have described the two types of architectures available in CGAs, competitive and cooperative, and their respective applications. We will now analyze in detail the two CGAs we used in our studies, one CGA in each class, namely CCGA (Cooperative Coevolutionary Genetic Algorithm) and LCGA (Loosely Coupled Genetic Algorithm), and study the existence of hybrid and dynamic variants of both of them.

2.2.4 CCGA: Cooperative Coevolutionary Genetic Algorithm

Cooperative (also called symbiotic) coevolutionary genetic algorithms (CCGA) involve a number of independently evolving species which together form complex structures, well-suited to solve a problem. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals.

Potter and De Jong [11] developed a model in which a number of populations explore different decompositions of the problem. In Potter's system, each species represents a subcomponent of a potential solution. Complete solutions are obtained by assembling representative members of each of the species (populations). The fitness of each individual depends on the quality of (some of) the complete solutions it participated in, thus measuring how well it cooperates to solve the problem. The evolution of each species is controlled by a separate, independent evolutionary algorithm. In the initial generation *(t=0)* individuals from a given subpopulation are matched with randomly chosen individuals from all other subpopulations. A fitness for each individual is evaluated, and the best individual in each subpopulation is found. The process of *cooperative coevolution* starts form the next generation *(t=1)*. For this purpose, in each generation a cycle of operations is repeated in a round-robin fashion. Only one current subpopulation is active in a cycle, while the other subpopulations are frozen. All individuals from the active subpopulation are matched with the best values of frozen subpopulations. When the evolutionary process is completed a composition of the best individuals from each subpopulation represents a solution of a problem.

Potter and De Jong developed two different ways of evaluating the fitness of an individual, which are:

- credit assignment algorithm 1 (CCGA-1), with which the best individuals of each of the other subpopulations are used to evaluate the fitness of an individual in a subpopulation, as described previously;
- credit assignment algorithm 2 (CCGA-2), with which each individual in a subpopulation is evaluated by combining it both with the current best individual from each of the other subpopulations and with a random selection of individuals from each of the other subpopulations. The two resulting vectors are then applied to the objective function and the better of the two function values is returned as the fitness of the individual being evaluated.

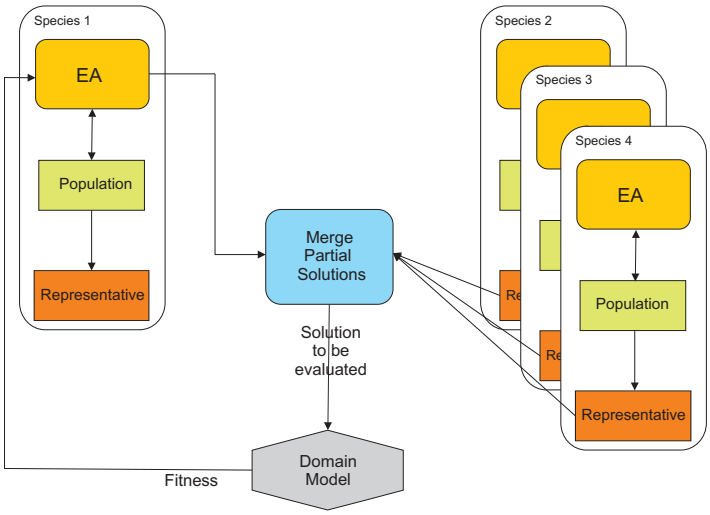


Figure 2.8: Potter and De Jong's CCGA architecture

Figure 2.8 shows the general architecture of Potter's cooperative coevolutionary framework, and the way each evolutionary algorithm computes the fitness of its individuals by combining them with selected representatives from the other species.

Algorithme 4 : CCGA

```
gen = 0
foreach species_s do
    Pop_s(gen) = randomly initialized population
    evaluate fitness of each individual in Pop_s(gen)
end
while termination condition = false do
    gen = gen + 1
    foreach species_s do
        select Pop_s(gen) from Pop_s(gen − 1) based on fitness
        apply genetic operators to Pop_s(gen)
        evaluate fitness of each individual in Pop_s(gen)
    end
end
```

2.2.4.1 Hybrid Cooperative Coevolutionary Genetic Algorithm

Hybridization of GAs for optimization purposes has been widely studied in the last years, combining the capacity of GAs to explore huge search spaces and find good regions of solutions with the exploitation power of local search algorithms. Hybrid coevolutionary algorithms are a new and promising alternative but contrary to hybrid GAs they are still quite unexplored.

Indeed, only Son and Baldwick in [25] proposed in 2004 a hybrid coevolutionary algorithm for Nash equilibrium search in games with local optima. They have incorporated a local hill-climbing algorithm to the basic coevolutionary algorithm, which is in fact a CCGA, so as to speed up convergence and fine tune control variables. To this end, they introduced the concept of "best rival matching and fine tuning" meaning that the chosen individual of one population is matched against the best strategies of the other populations in each generation and optimized using the hill climber. As experiment, they applied their hybrid algorithm on a transmission-constrained electricity market problem, showing that the hybrid version successfully avoided NE traps.

2.2.4.2 Adaptive Cooperative Coevolutionary Genetic Algorithm

In his PhD thesis [70] and later in [26], Potter studied the dynamic birth and death of subpopulations during the algorithms execution and consequently the emergence of an optimal number of these. He investigated this adaptive algorithm's capabilities on a neural network case study (for solving the two spirals problem). It provided good results on such simple problems but on more complex ones it will require a fitness function which more specifically allow the species to converge to different niches.

Another possible technique to improve the performance of GAs is to dynamically fine-tune their control parameters. Control parameters are fixed in classical GAs whereas self-adaptation occurs when the control of the mutation or crossover rates are represented within the chromosome as parameters. They consequently go through the same evolutionary processes as the problem parameters within the chromosome. This is a mean of providing the GA with more flexibility to conduct its search effectively and it allows the algorithm to adapt itself to the problem during the run.

For example, in [27], Ioro and Lee proposed a CCGA with self-adaptive control parameters. They used self-adaptive population size, crossover rate and mutation rate and compared the performance on these variations to CCGA-1 on the same test functions used by Potter (i.e. Rastrigin, Schwefel, Rosenbrock, Ackley and Griewangk). They observed some improvements of the solution when combining the adaptive sizing of the populations with either adaptive crossover rate or adaptive mutation rate.

Finally in [28], Cai and Peng introduced CCAGA (Cooperative Coevolutionary Adaptive Genetic Algorithm), in which the crossover rate is adapted (it is thus similar to one of Ioro's algorithm presented above). They used this algorithm for the optimization of a path-planning of multi-mobile robot systems.

2.2.5 LCGA: Competitive Coevolutionary Genetic Algorithm

The Loosely Coupled Genetic Algorithm (LCGA) [68] is a medium-level parallel and distributed coevolutionary algorithm exploring a paradigm of competitive coevolution motivated by non-cooperative

models of game theory [13].

The most widely used solution concept for non-cooperative games is the *Nash equilibrium point*. A Nash point is an N-tuple of actions, one for each player, such that anyone who deviates from it unilaterally cannot possibly improve his expected payoff u_k. If s_k denotes an action of the k-th player, then a Nash equilibrium point is an N-tuple $(s_0^*, s_1^*, \ldots, s_k^*, \ldots, s_{N-1}^*)$ such that

$$\begin{gathered} \forall s_k \neq s_k^*, k \in \{0, 1, \ldots, N-1\} : \\ u_k(s_0^*, s_1^*, \ldots, s_k^*, \ldots, s_{N-1}^*) \geq u_k(s_0^*, s_1^*, \ldots, s_k, \ldots, s_{N-1}^*) \end{gathered} \tag{2.1}$$

A Nash equilibrium point defines payoffs of all the players in the game. We are interested in some global measure of the payoffs received by players. This measure, called *price* of a game s, can be, e.g., the average payoff $\overline{u}(s)$ received by a player as a result of the combined actions $s = (s_0, s_1, \ldots, s_{N-1})$, i.e.,

$$\overline{u}(s) = \left(\sum_{k=0}^{N-1} u_k(s) \right) / N. \tag{2.2}$$

The question which arises immediately concerns the value of the criterion (2.2) at a Nash point. Unfortunately, this value can be very low.

Analyzing all possible action combinations in a game and evaluating their prices, i.e., the $\overline{u}(s)$ values, we can find action combinations characterized by a maximal price and we can call them *maximal price points*. Maximal price points are action combinations which maximize the global criterion (2.2), but they can be reached by players only if they are Nash points. A maximal price point usually is not a Nash point and the question which must be resolved is how to convert a maximal price point into a Nash point.

To solve this problem we introduce the concept of cooperation between players, by allowing them to share their payoffs in a game. We use two basic schemes of cooperation:

- *global cooperation:* sharing a payoff received by a player k with all players participating in the game, i.e., his payoff u_k is transformed into a new payoff w_k as

$$u_k \longrightarrow w_k = \overline{u}(s) \tag{2.3}$$

- *local cooperation:* sharing a payoff received by a player k with his neighbors in the game, i.e., his payoff is transformed as

$$u_k \longrightarrow w_k = \frac{\sum_{l \in N_k} u_l}{\max_{l \in \mathbf{N}} N_k + 1}, \tag{2.4}$$

  where u_l is the payoff of a neighbor l of the player k, N_k is the number of neighbors of the player k, and $\max_{l \in \mathbf{N}} N_k$ denotes the maximal number of neighbors in a game.

We are interested in a global behavior (in the sense of the criterion(2.2)) of a team of players taking part in iterated games. To evolve such a behavior we will use an evolutionary computation technique based on genetic algorithms (GA) [52].

For an optimization problem described by some function (a global criterion) of N variables, local chromosome structures are defined for each variable and local subpopulations are created for each of them. As opposed to known sequential and parallel EAs, the LCGA is assumed to work in a distributed environment described by locally defined functions. A problem to be solved is first analyzed in terms of possible decomposition and relations between subcomponents that are expressed by a communication graph G_{com}, aka graph of interaction. The objectives of this function decomposition and of the definition of the interaction graph are to minimize communications while still ensuring that the fact of reaching local optima for all different players (being a Nash equilibrium point) still leads to a global optimum of the initial function. This process has still to be done manually by taking into account information on the internal structure of the cost function, i.e. of the problem. The LCGA approach has been successfully applied for various problems including optimization of hard mathematical functions, multiprocessor mapping and scheduling. LCGA can be specified in the following way:

Algorithme 5 : LCGA

```
gen = 0
foreach player_s do
    Pop_s(gen) = randomly initialized population
    evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
end
while termination condition = false do
    gen = gen + 1
    foreach player_s do
        select Pop_s(gen) from Pop_s(gen - 1) based on fitness
        apply genetic operators to Pop_s(gen)
        evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
    end
end
```

After initializing subpopulations, corresponding sequences of operations are performed *in parallel* for each subpopulation, and repeated in each generation:
- For each individual in a subpopulation a number of n_i (n_i-number of neighbors of subpopulations $P^i()$) of *random tags* is assigned, and copies of individuals corresponding to these tags are sent to neighbor subpopulations, according to the interaction graph.
- Individuals in a subpopulation are matched with ones that arrived upon request from the neighbor subpopulations.
- Local fitness function of individuals from subpopulations is evaluated on the base of their values and

values of arrived tagged copies of individuals.
- Standard GA operators are applied locally in subpopulations.

Coevolving this way subpopulations compete to maximize their local functions. The process of local maximization is constrained by neighbor subpopulations, sharing the same variables. A final performance of the LCGA operated in a distributed environment is evaluated by some global criterion, usually as a sum of local function values in an equilibrium point. This global criterion is typically unknown for subpopulations (except the case when G_{com} is a fully connected graph), which evolve with their local criteria.

Contrary to the CCGA of De Jong, no hybridization of the competitive coevolutionary genetic algorithm LCGA was presented in the current literature. Consequently we will directly introduce in the next section the existing adaptive variant of the LCGA.

2.2.5.1 Adaptive Competitive Coevolutionary Genetic Algorithm

Seredynski et al. in [67] introduced an adaptive LCGA for distributed scheduling optimization. They proposed an approach to multiprocessor scheduling based on a multi-agent interpretation of the parallel program in which agents migrate in the topology of the parallel system environment, searching for an optimal allocation of program tasks into the processors. A collection of agents is assigned to tasks of the precedence task graph in a such way that one agent is assigned to one task. An agent A_k, associated with task k, can perform a number of migration actions S_k on the system graph . To evolve strategies for the migration of agents in the system graph, they use a LCGA. These local actions change the task allocation, thus influence the global optimization criterion.

2.2.6 Synthesis

Based on these descriptions, we can observe that the differentiation between these CGAs and between the variants of these CGAs (e.g. CCGA-1 and CCGA-2) lie in the following parameters:
- the topology of communications: complete graph for the CCGA, any topology for LCGA (depends on the problem decomposition)
- the interaction protocols: synchronous exchange from all to one for CCGA, asynchronous exchange for LCGA
- the exchanged information: best or best and random for CCGA, random individuals for LCGA
- their relation to the environment (i.e. to the optimization problem): in CCGA each subpopulation evaluates its solutions on the global problem, in LCGA a subpopulation evaluates its solutions on a subproblem (depending on the problem decomposition).

	CCGA	LCGA
Topology	Complete Graph	Any
Interaction protocols	Synchronous	Asynchronous
Exchanged information	best or best and random individuals	Random individuals
Problem view	Global	Local or Global

Figure 2.9: CCGA - LCGA comparison

It is therefore of interest for the user and for the system itself, to be able to manipulate these parameters so as to adapt these CGAs structure, interactions and exchanged information according to the optimization problem.

Few works tried to build new variants of these algorithms, either hybrid or adaptive.

Finally, as for CGAs in general, the applications of both CCGA and LCGA are mostly targeted to test problems and few works tried to benefit from CGAs on real-world problems.

According to this first state of the art, the work presented in this dissertation will investigate how to address the three issues aforementioned:

- by providing a high-level model allowing to specify CGAs in a self-expressive manner, to both the designer and the system, in terms of the algorithms' topologies, interactions and exchanged information
- by taking advantage of this model to build new hybrid and dynamic CGA variants,
- by applying those CGAs, i.e. existing and new ones, on real-world optimization problems.

2.3 Frameworks for Distributed and Parallel Evolutionary Computation

In the field of parallel evolutionary algorithms (PEAs), many implementations have been realized since the late eighties, ranging from simple libraries to complex frameworks. The existing literature provides several surveys of these implementations, from which we can cite [7] and [77].

The objective of this section is to provide an overview of the existing frameworks dedicated to PEAs classified according to the paradigm they use, i.e. object or agent. We will not relate the whole history of PEAs implementations since it is well studied in the papers previously referenced. We will only emphasize the current PEA frameworks which are commonly considered as outstanding which are Dream, ECJ, ParadisEO, JDEAL and MALLBA for the object oriented platforms and MAGMA and MAS-DGA for the agent oriented ones.

2.3.1 Object Oriented PEAs platforms

- ParadisEO (PARAllel and DIStributed Evolving Objects) [18] allows the use of PEAs with either an island-model or a master-slave configuration. It is not limited to EC as it can also perform local search algorithms. The framework is coded in C++ using MPI/PVM for message passing and the pthread library for multi-threading. Paradiseo is based on a modular architecture, as shown in Figure 2.10, in which Paradiseo-EO allows the use of population-based metaheuristics (like GAs), Paradiseo-MO allows the use of solution-based metaheuristics (e.g. Hill Climbing), Paradiseo-MOEO permits multi-objective optimization (e.g. NSGA-II) and Paradiseo-PEO allows to utilize parallel, hybrid and distributed metaheuristics (e.g. island model).

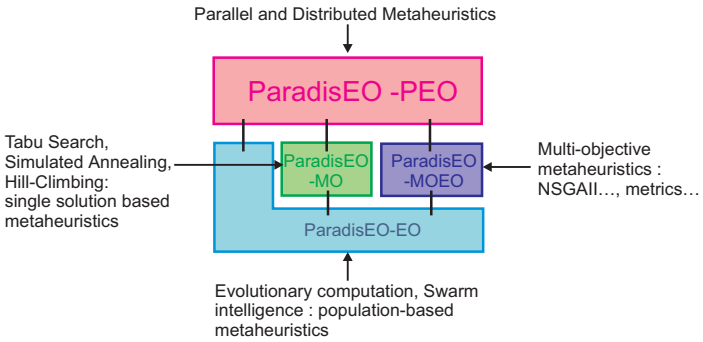


Figure 2.10: Paradiseo: A Module-Based Architecture

- DREAM (Distributed Resource Evolutionary Algorithm Machine) [20] is a peer-to-peer software based on the island model implemented in Java. It considers a virtual pool of distributed computing resources, in which each node evolves its own population. DREAM's architecture is split into five modules (see Figure 2.11), GUIDE (Graphic User Interface for DREAM Experiments) which allows to build distributed evolutionary algorithms using a fully graphical interface, EASE (EAsy Specification of Evolutionary Algorithms) which provides a high level textural language to program distributed evolutionary algorithms, JEO (Java Evolutionary Object) which is a Java library for evolutionary computing, DRM (Distributed Resource Machine) API and finally the DREAM console, primary tool for managing a computer connected to a DRM.

  Each module can be used either in standalone mode or in integrated mode, which allows the user to choose his specification level according to his skill level. DREAM is targeted toward Wide Area Networks (WAN) where communication costs are high. It assumes an application which is massively parallelizable, asynchronous and robust (i.e. its success does not depend on the success of any sub-process), that requires little communication between sub-processes, and has large resource requirements.

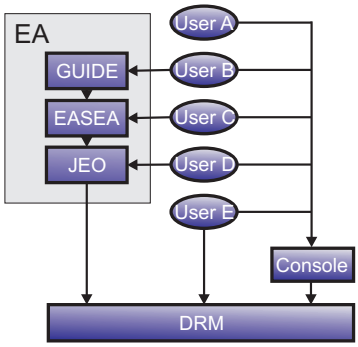


Figure 2.11: DREAM: Overall Architecture

- ECJ [21] is an open Java-based framework for evolutionary computation and genetic programming. It uses TCP/IP and multi-threading to efficiently parallelize its algorithms. The distribution can be based either on the island model or on the master-slave strategies. Its distribution is less sophisticated than in DREAM but it is sufficient for a deployment on a Local Area Network (LAN). ECJ also provides some additional features like multi-objective optimization, logging possibilities and check-pointing, etc.

- JDEAL (Java Distributed Evolutionary Algorithms Library) [78] is another portable Java framework for EAs which integrates its own implementation of genetic algorithms and evolution strategies. Both local and MasterSlave parallel/distributed models are available. Its ease of use is ensured through a documentation and a tutorial. JDEAL also includes support tools such as automatic generation of execution statistics and check-pointing.

- The MALLBA (MAlaga + La Laguna + BArcelona) [19] framework is a library of skeletons for combinatorial optimization including exact, heuristic and hybrid methods. Skeletons are algorithmic units that, in a template-like manner, implement generic algorithms. The skeletons are implemented using C++. MALLBA is targeted to sequential computers, LANs and WANs environments on which it can be transparently deployed. Communications are based on NetStream, a flexible and simple OOP message passing service, upon MPI. Portability is ensured by the utilization of the C++ language and standards such as MPI.

2.3.2 Agent Oriented PEAs platforms

Only two frameworks, MAGMA (MultiAGgent Architecture for Metaheuristics) and MAS-DGA (Multi-Agent System for Distributed Genetic Algorithms) [79], use a multi-agent architecture for metaheuristics algorithms.

- MAGMA (MultiAGgent Architecture for Metaheuristics), introduced in Roli's Phd thesis [80] and later in [81], consists in a multi-level architecture where each level contains one or several

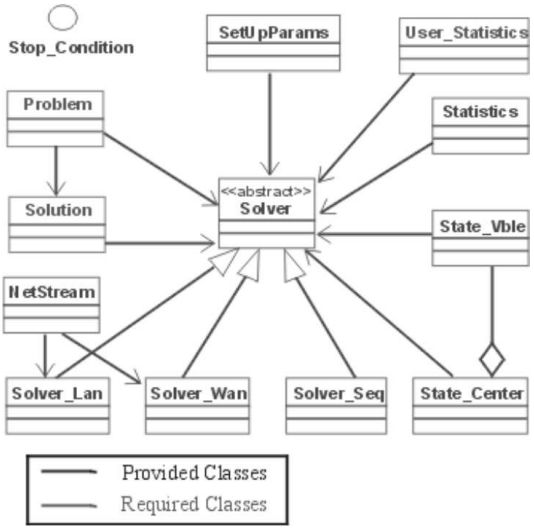


Figure 2.12: MALLBA

specialized agents implementing an algorithm. Some of these levels, and consequently agents propose several metaheuristics, including GAs. Figure 2.13 presents an example of the usage of MAGMA's multi-level architecture to model a Memetic Algorithm (i.e. a GA combined with a local search algorithm). MAGMA has been introduced as a conceptual framework to model various metaheuristics and a partial parallel implementation has been proposed, with which experimental results on the Maximum Satisfiability Problem (MAXSAT) using Ant Colony Optimization (ACO) and Iterated Local Search (ILS). However no implementation of GAs in MAGMA have been referenced up to now.

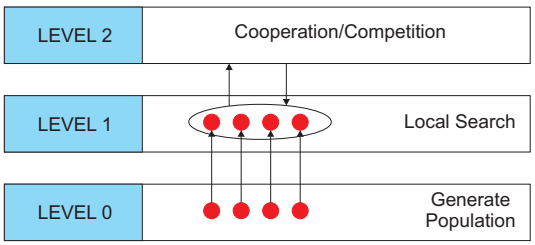


Figure 2.13: MAGMA Multi-Level Architecture Example: Memetic Algorithm (MA)

- In MAS-DGA (Multi-Agent System for Distributed Genetic Algorithms) [79], each basic GA is encapsulated into an agent, an autonomous entity that must keep knowledge of the search, learning, or optimization problem it should operate on. Agents should be coordinated through a set of rules stipulating the topological and communication (migration) aspects, and these rules may be fixed a priori or set in run-time via a coordination entity (meta-agent). MAS-DGA was used for experimenting and adaptive migration policy in a coarse-grain PGA (i.e. island

Framework	Paradigm	Algorithms	Parallelization	Implementation
ParadisEO	Object	GA, SA, ES, ACO, PSO, local search, hybrid (low and high level)	master-slave, island and cellular models	C++ (MPI/PVM)
DREAM	Object	GA, ES	island model	Java
ECJ	Object	GA, PSO, DE, GP	master-slave, island model	JAVA (sockets)
JDEAL	Object	GA, ES	master-slave	JAVA (sockets)
MALLBA	Object	GA, SA, ES, ACO, PSO, local search, hybrid	master-slave, island model	C++ (MPI)
MAGMA	Agent	GA, ACO, PSO, local search, hybrid	island model	—
MAS-DGA	Agent	GA	island model	—

Figure 2.14: Evolutionary algorithms frameworks comparison

model). No details concerning its multi-agent model or implementation are given in the single paper mentioning MAS-DGA.

2.3.3 Synthesis

As can be seen in this state of the art, these frameworks allow the use of many different PEAs, but none of these allow the use of coevolutionary GAs like CCGA or LCGA. A few of them propose some coevolutionary algorithms, like PAradiseo or ECJ, but they are different from De Jong's and Seredynski's algorithms. Indeed they consist in hybrid coevolutions for Paradiseo (as described in [82]) or in simple coevolution for ECJ. Additionally, most of these frameworks are implemented using the object paradigm. They do not provide any high level modeling language describing the structure or the interactions of the PEAs they propose.

Only two platforms use the agent paradigm, MAGMA providing a more complete framework, i.e. a richer model and implementation possibilities. However very little information is available concerning their implementation and their configuration/usage. Finally there is no available version of these implementations.

To conclude, there is currently no framework dedicated to CGAs. Only a few PEAs platforms allow the use of some CGAs but not the most well-known ones like CCGA.

Based on these observations, we chose to orientate our research on the development of a new platform dedicated to CGAs. The latter will have to use a paradigm which will allow taking into account the high-level model we first discussed in 2.2.6.

2.4 Conclusion

The aim of this chapter was to introduce coevolutionary genetic algorithms (CGAs), their different architectures (competitive and cooperative), their available variants (hybrid and dynamic) and their application domains. It appears that many different problems were tackled using CGAs but only few

are real-world problems. We provided some more detailed description of the two CGAs we used and extended in our research works: CCGA and LCGA.

We observed that the differentiation between these CGAs and between the variants of these CGAs (e.g. CCGA-1 and CCGA-2) lie in their topology of communication, in their interaction protocols, in the type of information they exchange and in their relation to the environment (i.e. to the optimization problem). It is therefore of interest for the user and for the system itself, to be able to manipulate these parameters so as to adapt these CGAs structure, interactions and exchanged information according to the optimization problem.

Based on these observations, we studied the most well known available frameworks for Parallel and distributed Evolutionary Algorithms (PEAs) and came to the first conclusion that only a few of these permit the use of a limited number of CGAs. They additionally do not take advantage of high-level models, methodologies which would allow to explicitly define the algorithms structures and interactions and consequently to make them easily manipulable by the user or by the systems themselves.

Chapter 3

Multi-Agent Organizations and Adaptation

Contents

In the previous chapter, we have demonstrated the following needs for building CGAs:

- make the structure (i.e. the topology) of the CGAs explicit and modifiable by the user and/or by the system
- make the interactions between the CGAs components (i.e. communications between subpopulations) explicit and modifiable by the user and/or by the system
- make explicit the interactions of the system with its environment which can change in time (i.e. for dynamic optimization problems)

We have also shown that the most well known available frameworks for Parallel and distributed Evolutionary Algorithms (PEAs) do not allow the use of common CGAs and mostly work on a "black box" model, hiding the structure, the interactions or the relations of the system to the optimization problem. We therefore investigate the use of the agent paradigm (as opposed to the object paradigm) to model an application by taking into account those different dimensions.

In this aim, following chapter provides an introduction to the agent paradigm and multi-agent systems. We then introduce the notion of organization in multi-agent systems and study the existing organizational models and their limitations when facing some system modifications. Finally, we study how reorganizational models can handle online changes in a multi-agent system, issue that we face when developing a new dynamic CGA inside our framework.

3.1 Definitions

3.1.1 Agent

As Michael J. Wooldridge mentioned in [83], there is no universally accepted definition of the term agent and there is a good deal of ongoing debate and controversy on this subject. One of the reasons is that many different communities revendicate this term but having some different problematic and thus different perspectives and techniques.

In this dissertation, we will consider the definition provided by Wooldridge and Jennings in [83], in which an agent is "a hardware or (more usually) a software-based computer system that enjoys the following properties:

- autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it.
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking initiative.

- social ability: agents interact with other agents (and possibly humans) via some communication languages.

Multiple kinds of agent architectures exist, from very basic to very complex (see 3.2.1 for more details). On basic ones we can find some common characteristics like autonomy (independence), persistence (long-liveness), monitoring of the environment, communication and collaboration with other agents and/or the user. More "intelligent" agents possess higher-level abilities, such as mobility, decision-making, and the ability to learn.

3.1.2 Multi-Agent System

As mentioned in [84], the definition of a Multi-Agent System (MAS) is more immediate than the agent one: "a multi-agent system is an organized collection of agents". This means that in a multi-agent system, one or several organizations exist which provide a structure for the cohabitation and the collaborative work of agents (definition of the different roles, resources sharing, dependencies between tasks, coordination protocols, conflicts solving, etc.). In a single system, it usually exists several organizations and a single agent can belong to multiple ones. Organizations in the real-world can range from animal organizations (e.g. ant colonies) to social or economical organizations (e.g. community or company). Depending on the needs, the agents' behaviors will be more or less complex and the organization more or less adaptive.

A key concept for multi-agent systems is the balance (and complementarity) between autonomy and organization. Agents are usually located in an environment (for instance geographical) which contains passive entities manipulated by the agents (e.g. data, resources, concrete objects, etc.). Each agent has only a partial view on the system (i.e. the other agents) and on its environment. Consequently, a multi-agent system is intrinsically decentralized.

Multi-agent systems have been widely applied for electronic commerce, collective robots, process control or complex phenomenon simulation. A standardization of agents and their interoperability has been initiated by some industrial leaders in telecommunication through the Foundation for Intelligent Physical Agents (FIPA)[1] [85].

3.2 Multi-Agent Models

In the following section, we describe the different models manipulated within a MAS by using the vowel approach AEIO (Agent, Environment, Interaction, Organization) introduced by Demazeau in [86].

[1] Foundation for Intelligent Physical Agents: http://www.fipa.org

Agents deal with the models (or architectures) used for the active part of the agent, from a simple automata to a complex knowledge based system. *Environments* are the places where the agents are located. *Interactions* concern the infrastructures, the languages and the interaction protocols between agents, from simple physical interactions to complex communicative acts. *Organizations* structure the agents in groups, hierarchies, relations, etc.

3.2.1 Agent

Many different agent architectures have been proposed in the literature. Surveys of these architectures and their applications were introduced in [87] [88] [89]. Boissier[1] [2] also proposed a classification based on a two-dimensional grid.

The first dimension characterizes the internal control of the agent. Agents can be first reactive, deliberative or hybrid. *Reactive agents* have a simple stimulus-response functioning loop. They react to the environment's evolution but have no representation of it, of the other agents, etc. They additionally have no history (reference to the past) and no planning (reference to the future). *Deliberative agents* add a deliberative function in the functioning loop of an agent. Using a representation of the environment, of the agents and using history and planning capabilities, the agent can choose the "right" action . *Hybrid agents* are thus both reactive and deliberative, depending on the situation.

The second dimension classifies the agents according to their capacities or competencies in relation with the MAS models. Agents can be individual agents, social agents or organized agents. *Individual agents* reason about themselves and about the system's environment (i.e. the common space of the agents, see definition in 3.2.2). The environment has thus to be modeled (see 3.2.2). *Social agents* also reason on themselves, on the environment, but also on the interactions with the other agents. Interactions have thus to be modeled, usually using exchanges of messages (e.g. interaction protocols, see 3.2.3). Finally, *organized agents* reason about themselves, the environment, the interactions and the organizational structures imposing these interactions.

Table 3.1 sorts some of the existing agent architectures according to the aforementioned two dimensions.

	Reactive Agents	**Deliberative Agents**	**Hybrid Agents**
Individuals Agents	Subsumption Architecture [90], MANTA [91]	BDI (Belief, Desire, Intention) [92]; PRS	Touring Machines [93]
Social Agents	SAM [94]	AOP [95]	InteRRaP [96]
Organized Agents		B-DOING [97]	

Table 3.1: Agent Architectures Classification from [2]

[1] Lecture on MAS: http://www.emse.fr/∼boissier/enseignement/smacourant/index.html

3.2.2 Environment

A distinction between the environment of a MAS and the environment of an agent has to be done. Indeed, the environment of a MAS corresponds to a common space for the agents of the system while the environment of an agent corresponds to the environment of the MAS plus the other agents belonging to the system.

Depending on the MAS application domain the environment has different modelings. In simulation, the environment of a MAS is composed of active and passive objects which can be manipulated by the agents. On the contrary, in the case of collaborative systems, the environment is constituted of information which can not be controlled by the agents (e.g. the Web). Finally, in distributed problems solving, the MAS environment corresponds to the problem's data and the context in which the rules encoded in the agents, the interactions and the organizations are updated. The environment can be coupled with the agents (reacting to their actions) or decoupled (agents do not modify the environment).

As for the environment definition, actions on the environment depend on the application domain. In the case of simulations, an action can be defined as the mechanism implying a modification of the physical environment or by the reaction (modification) of the environment due to the agent's action. For problem solving, agents evolve according to the environment and for collaborative systems agents do not have actions on the environment.

The same way, concerning the perception of the environment, in the case of simulation it is done through a sensor, while for problem solving it is a selection mechanism of a point of view on the problem and for collaborative systems it is an interpretation of external data.

The environment can be classified according to the following set of properties proposed by Russel and Norvig [98]:
- Accessible/Inaccessible: if an agent has sensors which allow him to observe the complete state of the environment and choose an action accordingly, then the environment is accessible.
- Deterministic/Non-deterministic: the environment is deterministic if the next state of the environment is determined by its current state and the actions selected by the agents.
- Episodic/Non-episodic: In an episodic environment, the agents experience is divided into "episodes". Each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself, because subsequent episodes do not depend on what actions occur in previous episodes.
- Static/Dynamic: the environment is static if it does not change while the agent is deliberating.
- Discrete/Continuous: the environment is discrete if agents have a limited number of perceptions and

actions.
- With/Without rational opponents

3.2.3 Interaction

Communications in MAS are the foundation of agents interactions and organization. We can distinguish two communication modes, the indirect communication which is a signal communication through the environment and the direct communication which consists in messages exchanges between agents. In the following we will only focus on the last issue: communication.

The inter-agent communication can be modeled by two approaches. In the first one the interaction is emergent, a communication plan is built according to the agent knowledge and goals. In the second approach, the interaction respects a set of messages specified in advance called *interaction protocol*. In this work we will only consider interaction protocols.

The origins of multi-agent communicative languages come from the speech acts theory introduced by Austin in [99]. He introduced three types of speech acts, the locutory act (the act of saying something), the illocutory act (realized action when saying something, e.g. request) and the perlocutory act (the obtained effect, e.g. to convince). These works are the origins of the multi-agent communication languages we introduce hereafter.

Communication languages allow to facilitate the exchange and interpretation of messages as well as the agents interoperability. Finin in [100] specified a set of properties the communication languages should comply to. They should have a declarative form, a simple and readable syntax, distinguish between communication language (describing the speech acts) and content language (expressing the contained information), have a formalized language semantic and have an efficient implementation.

The first language introduced is KQML (Knowledge Query and Manipulation Language) [101] which was initially developed to exchange knowledge and information between knowledge-based systems. It was then used to describe the messages exchanged between agents.

The second language, FIPA-ACL (FIPA Agent Communication Language) [102], was proposed in the context of a standardization effort of the FIPA (Foundation of Intelligent Physical Agents). FIPA-ACL is an extension of KQML. FIPA-ACL is based on twenty one speech acts expressed by performatives grouped by functionalities. A message can contain one part or all the elements presented in table 3.2. Mandatory elements to transmit a message vary according to the situation. If one agent does not recognize or can not treat one or several elements, he can reply with a *Not-understood* message.

Parameter	Category of Parameters
performative	Type of the communicative act
sender	Sender of the message
receiver	Receiver of the message
reply-to	Participant in the communication act
content	Content of message
language	Language in which the content is expressed
encoding	Description of the message encoding mode
ontology	Name of the ontology used to give a sense to the content terms
protocol	Name of the interaction protocol
conversation-id	Identifier of the conversation
reply-with	Identifier of the message, for a future reference
in-reply-to	Reference of the message to which the agent is answering
reply-by	Delay to answer the message

Table 3.2: Elements of a FIPA-ACL message

Communication languages are utilized to specify interaction protocols. Those interaction protocols are used by agents to govern their interactions. An interaction protocol is built on four main notions. Firstly, an interaction protocol is a pattern of interaction, independently to the application context. Secondly, each interaction protocol has a goal. Thirdly, an interaction protocol involves two or more agents, each of them playing a role which allow to identify it. Fourthly, an interaction protocol defines the rules which govern an interaction. These rules define the sequence of messages and the actions the protocol refers to.

Several interaction protocols' specification languages have been developed. Among them we can cite transition nets (Finite State Machines [103], Petri Nets [104], logic based specifications [105], AUML (Agent UML) [106]. The latter is an extension of UML (Unified Modeling Language) sequence diagrams and is used by the FIPA to specify its specific interaction protocols (e.g. fipa-request, fipa-contract-net, etc.).

3.2.4 Organization

The definition of the organization depends on the type of MAS. However we can generally say that the organization can be viewed as the topology of a group which allow to specialize the agents according to their skills.

The organization in a MAS is composed of an *organizational structure* (OS) which describes the organization in terms of roles, tasks, responsibilities, and an *organizational entity* (OE) which instantiates this structure. The organizational entity represents the affectation of the agents to the roles described in the OS.

An organization can be:
- Explicit (see 3.3.2): roles are defined a priori. The designer knows the organizational structure.
- Emergent (see 3.4.1): roles are observed a posteriori. No organizational structure is known by the designer.
- Static: the structure or the organizational entities are fixed a priori and can not evolve during the simulation.
- Dynamic: the structure or the organizational entities can evolve during the simulation.

A more detailed description of organizations in MAS is provided in the next section (3.3).

3.3 Organizations in Multi-Agent Systems

3.3.1 Definition

As for the agent concept, there are several definitions of what an organization exactly means. Indeed, the word "organization" is a complex word that has several meanings. In [107], Gasser proposed the definition of organization to which we subscribe: *An organization provides a framework for activity and interaction through the definition of roles, behavioral expectations and authority relationships (e.g. control)* . Horling in [108] uses a similar notion by defining the organizational structure of a multi-agent system as *the rules which define the roles agents play and the manners in which they interact with other agents in the system.* For Dignum in [109] the organization of a Multi-Agent System (MAS) can be seen as a set of constraints that a group of agents adopts in order to easily achieve their social purposes. These definitions are rather general and do not provide any clue on how to design organizations. In [110] Jennings and Wooldridge propose a more practical definition: *We view an organization as a collection of roles, that stand in certain relationships to one another, and that take part in systematic institutionalized patterns of interactions with other roles.* However, in [23] Ferber pointed out such a definition lacks an important feature of organization, i.e., partitioning, a tool to partition the system. According to him, organizations are structured as aggregates of several partitions which may overlap and each partition may itself be decomposed into sub-partitions.

In our opinion it is worth considering a general definition since there is still a lot of interpretations of what an organization is and it would be premature to give a restrictive definition. Consequently we will keep as reference an extended definition of the definition given in [111] where an organization is a set of constraints (structures, norms and patterns) found in a social context that shape the actions and interactions of agents to achieve a social goal.

There exists two possible ways of obtaining an organization in a MAS, either with a bottom-up approach or with a top-down approach. Contrary to the top-down approach, the bottom-up one does not manipulate any organizational model defined a priori but it utilizes some interaction capabilities to dynamically create and adapt the MAS organizations. The following section provides details on such

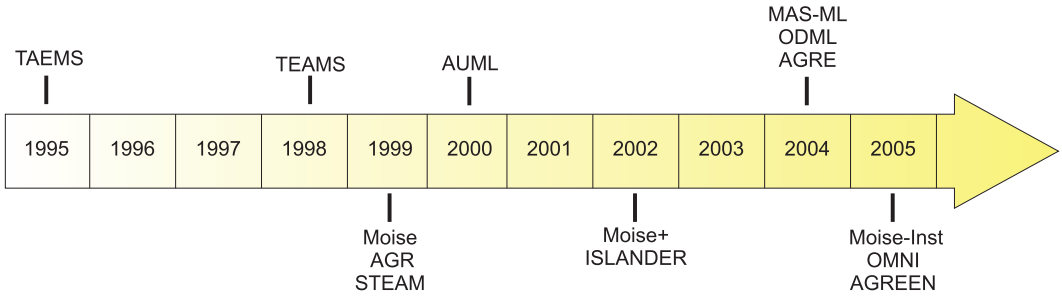


Figure 3.1: Organization Models Chronology

top-down organizational models while the intrinsically dynamic bottom-up approach will be discussed in section 3.4.1.

3.3.2 Organizational Models

An organizational model allows to represent the social organization within a multi-agent system. Multiple organizational models have been proposed, chronologically we can cite TAEMS (1995) [112], TEAMS (1998) [113], MOISE (1999) [114], AGR (1999) [23], STEAM (1999) [115], AUML (2000) [116], ISLANDER (2002) [117], Moise+ [118], MAS-ML (2004) [119], ODML (2004) [120], AGRE (2004) [121], Moise-Inst (2005) [24], OMNI (2005) [122] and finally AGREEN (2005) [123].

An interesting survey was proposed on some of these organizational models by Coutinho et al. in [111], however in the coming survey we will not describe all of these model but only AGR, Moise+, OMNI and ISLANDER are discussed in detail.

3.3.2.1 AGR

The AGR (Agent, Group, Role) organizational model [23] is an evolution of the AALAADIN model [124]. In AGR, agents play roles in different groups. An agent is an active, communicating entity playing (one or multiple) roles within (one or several) groups. No constraints are placed upon the architecture of an agent or about its mental capabilities (i.e. it can be reactive like an ant or as clever as a human). A group is a set of agents sharing some common characteristic. A group is used as a context for a pattern of activities, and is used for partitioning organizations. Agents may communicate if and only if they belong to the same group. A role is the abstract representation of a functional position of an agent in a group. An agent must play a role in a group, but an agent may play several roles. Roles are local to groups, and a role must be requested by an agent. A role may be played by several agents.

The main characteristic of the AGR model is its minimalist structure-based view of organizations. In an AGR model an organization is specified as a role-group structure imposed on the agents. AGR also says that agents can have their joint behavior orchestrated by interaction protocols, but the nature

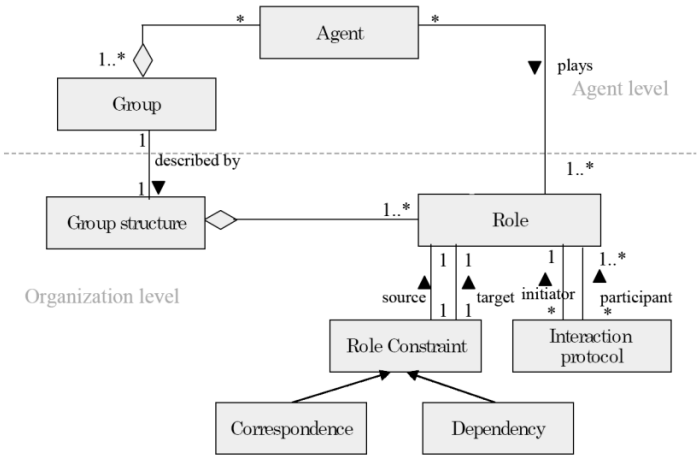


Figure 3.2: AGR Meta-Model

and the primitives to describe such protocols are left open. Some recent works provided extensions to AGR: AGRE [121] takes into account the environment in the model and AGREEN [123] tries to integrate both environment and norms within the multi-agent organizations. To conclude, AGR does not provide any definition of the functionality of the organizational structure and consequently the view on the organization is only partial. Additionally AGR does not allow inheritance or hierarchy between roles. The organization description is abstract, i.e. there is no description language which can be used by the agents, and is therefore not accessible to the agents once instantiated.

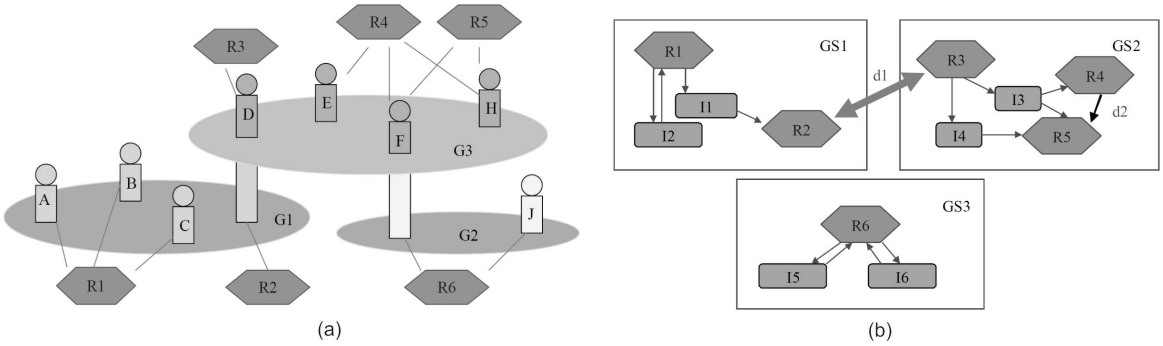


Figure 3.3: AGR: (a) Concrete Organization (Cheeseboard diagram), (b) Organizational Structure

3.3.2.2 MOISE+

MOISE+ (Model of Organization for multI-agent SystEms) [125] [126] is an organizational model which defines constraints on agents using three specifications: the structural, the functional and the deontic specifications. The Structural Specification (SS) is built on the concepts of roles, links between roles, and groups. It is graphically represented as shown in Figure 3.4(a). Roles can be hierarchically

organized and can be linked between each others using different links. The Functional Specification (FS) is built on the concepts of social scheme, plan, goal and mission. As illustrated in Figure 3.4(b), a social scheme is composed of goals, plans and missions and represents the function of one or several agents within the organization. Finally, the Deontic Specification (DS) connects the two previous specifications (one role with one mission) using some deontic operators (Obligation, Permission or Interdiction). The specification of these three dimensions forms an Organizational Specification (OS) while the instantiation of these three dimensions using agents playing roles and thus constrained by the organization represent an Organizational Entity (OE). Gateau in [24] introduced an extension to Moise+ called Moise-Inst. It keeps the structural and functional specifications unchanged, but instead of the deontic specification, Moise-Inst provides the contextual and normative specifications (CS and NS).

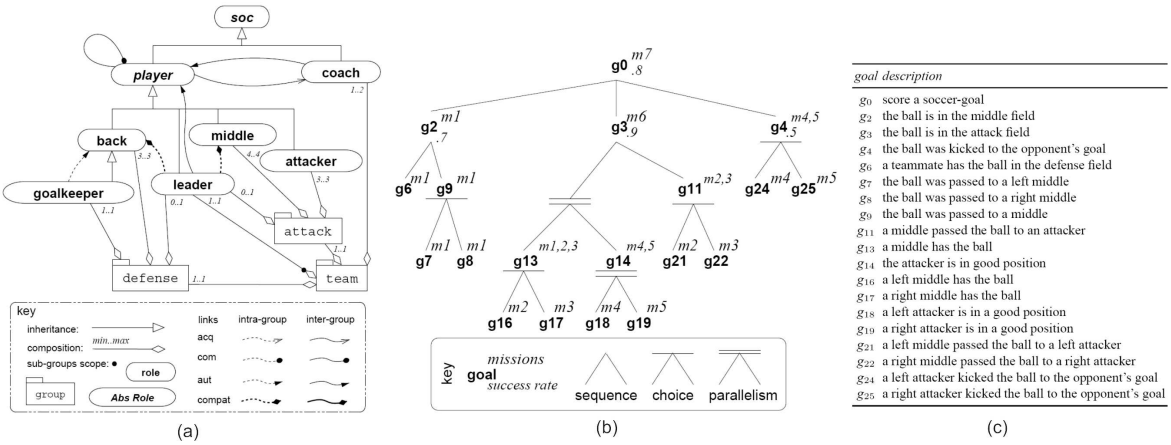


Figure 3.4: Moise+: (a) Structural Specification (SS), (b) Functional Specification (FS), (c) Deontic Specification (DS)

If we compare MOISE+ with AGR, we can see that AGR only corresponds to the structural dimension of MOISE+. The structural dimension of MOISE+ also extends the concepts found in AGR by providing abstract roles, inheritance and composition between roles, and communication and authority links. Up to now it is one of the most complete structural specification. Contrary to AGR, Moise+ does not directly specify the interactions between agents since no protocol can be defined but only constraints on communications through the use of links. Finally, using three different specifications allow a high flexibility to the model.

3.3.2.3 OMNI

OMNI (Organizational Model for Normative Institutions) [122] [127] [128] is an integrated framework for norms, structure, interaction and ontologies for modeling organizations in MAS. Its is a unification of two other models: the OperA [129] which brings the organizational dimension and the HarmonIA

framework [130] which brings a normative dimension.

As illustrated in Figure 3.5, the OMNI model can be represented as a matrix with on the one hand a division into three dimensions (organizational, ontological, normative) and on the other hand a division into levels (abstract, concrete, implementation).

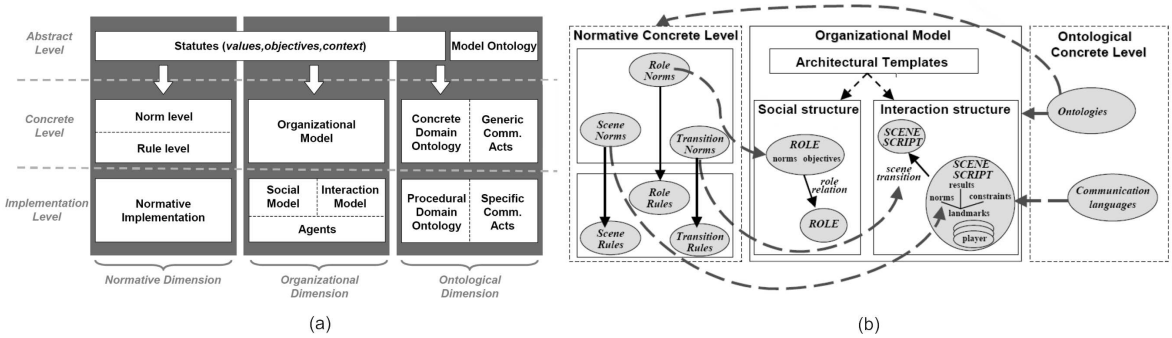


Figure 3.5: OMNI: (a) Levels and Dimensions, (b) Concrete Level Details

The concrete organizational model is composed of a role structure and an interaction structure which will be implemented in two models. The social model specifies the roles played by the agents and the interaction model describes the interactions between the agents. The role structure is a dependence tree of roles and each role is associated to one objective (i.e. the functional specification is integrated in the structural specification). The interaction structure allows to define scenes linked between each other with transitions. Finally norms are used to specify roles or scene's scripts. They are linked to the elements of the organizational dimension.

However, no system function, plan or execution scheme is defined with OMNI. Only some objectives and sub-objectives can be used as functional structure. These objectives can be found in the role and interaction structures and are consequently dependant from the specification of the other dimensions.

3.3.2.4 ISLANDER

ISLANDER is an IDL (Institution Definition Language), a declarative language for specifying electronic institutions [117]. This language is based on XML and is a syntax defining Electronic Institutions represented as a dialogic system allowing messages exchange. These interactions are structured through groups of agents called scenes which follow explicitly defined protocols. In ISLANDER, electronic institutions are composed of four basic elements: (1) a dialogic framework (DF), (2) scenes, (3) performative structure (PS), and (4) norms (see Figure 3.6)

.

The dialogic framework contains the elements for the construction of illocutions that agents can exchange. The goal is to allow agents sharing the same dialogic framework to exchange some knowledge. It is composed of one ontology, a set of definitions of types and functions, illocutions and roles. Each role defines a pattern of behavior within the institution and any agent within an institution is required to adopt some of them. It is possible to specify relations between roles using hierarchical links (i.e. inheritance) and static separation of duties (ssd) links (specifying that one agent can play both roles). A scene is a collection of agents playing different roles in interaction with each other in order to realize a given activity. Every scene follow a well-defined communication protocol. The communication protocol of a scene is stated in terms of possible dialogic interaction between roles instead of agents. The communication protocol dictates for each agent role within a scene what can be said, by whom, to whom, and when.

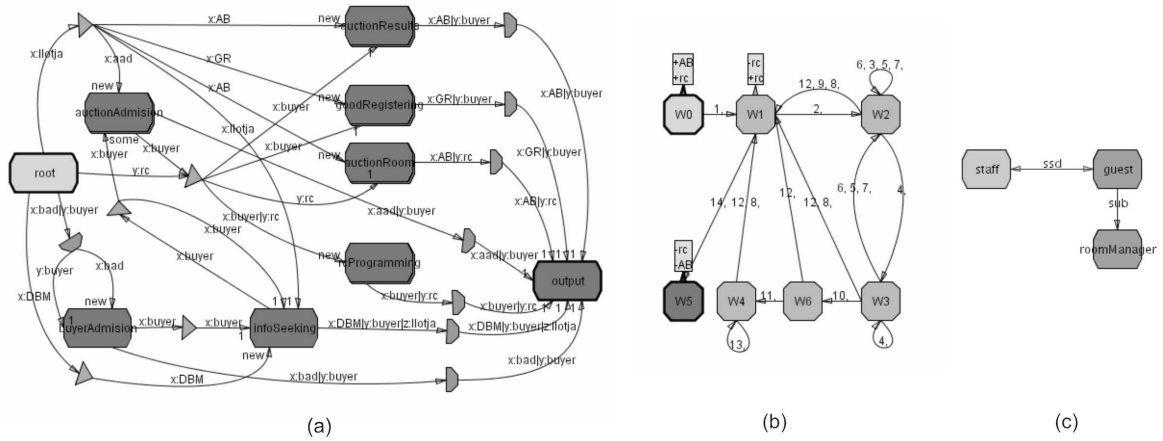


Figure 3.6: ISLANDER: (a) Performative Structure (PS), (b) Scene, (c) Roles

The performative structure defines a set of scenes linked by transitions. A scene is a group of agents which follow some specific communication protocol. Transitions between scenes determine how agents can go from one scene to another according to the role they play given some constraints (e.g. a conversation can start in the new scene).

Norms allow to capture the consequences of agents' actions using obligations, commitments and rights.

Compared to Moise+, the structural specification is minimal, indeed it is only possible to define roles, sub-roles and compatibility links between these roles. Actually it is somehow similar to OMNI with this role dependence tree. Norms are also expressed in a simpler way compared to Moise+ (i.e. they can only be obligations) but Moise+ does not provide any interaction definition. Norms are only applied on interactions while and scene transitions while they are applied on missions in Moise+ and

role interactions in OMNI.

3.3.3 Synthesis

The following table is an extension of the table proposed by Coutinho in [111].

	Modeling Dimensions			
Model	**Structural**	**Dialogical**	**Functional**	**Normative**
AGR	++	+	-	-
AGRE	++	+	-	-
AGREEN	++	+	-	+
MOISE+	+++	-	++	+
MOISE-Inst	+++	-	++	++
OMNI	++	+++	+	+++
ISLANDER	+	+++	-	++

Table 3.3: Organizational models comparison. A model having more (+) in a given modeling dimension means that the model offers more concepts and elements in the given dimension. A (-) means that the model does not support modeling in the dimension. This table is only an approximation to give the reader a feeling of the relative expression power of each organizational model

The different organizational models we studied in this section provide ways of describing static organizations. However, as soon as the organization becomes dynamic (i.e. changing in time), such models are unable to deal with adaptation requirements. That is why in the coming section we study how reorganization issues can be tackled in a MAS.

3.4 Adaptation of Multi-Agent Organizations

By adaptation of Multi-Agent Organizations we mean the set of transformations to which the organization in an open multi-agent system can be submitted during the system's functioning, due to the mutual influence of: its functional constraints, the changes in its environment, the entrance and the departure of agents in its structure, or the conceiver's or the user's intervention.

Having a statically defined organization as introduced in 3.3.2 is by default the simplest option but it becomes intractable as soon as the system changes in time. Indeed, as Horling mentions in [108] if elements of the multi-agent system are dynamic (environment, organizational goals, member agents) the initial organization will become inefficient, having for instance deactivated, compromised existing agents or not effectively used new agents.

Hübner et al. in [126] express the same problem by the tradeoff that has to be found between collective constraints and agent autonomy. They additionally mention that in the case of a static organization, all the experience and information collected about the organization by the agents and that

could be used to adapt it is lost. In other words agents loose their autonomy regarding the organization.

For Mathieu [131] the basic problem lies in the acquaintance creation, i.e. how agents can optimally communicate with each other in a dynamic system. Like Horling, Mathieu also cites as a problem the skills distribution over the agents that should prevent (if possible) situations like one single agent becoming an overloaded resource. The final drawback he refers to is the incapacity to share knowledge that would save communications. Indeed if one agent constantly refers to another one it might save communications and time if the customer acquires the concerned service.

The organization is thus a set of assumptions that has to be able to change so as to keep a viable system.

As we already mentioned in 3.2.4, organizations in MAS can be obtained using a top-down or a bottom-up approach. The bottom-up one does not manipulate any organizational model defined a priori but it utilizes some interaction capabilities to dynamically create and adapt the MAS organizations. In the literature, this is usually refereed as self-organizing MAS or Adaptive MAS (AAMAS) constituted of Adaptive Agents (represented in Figure 3.7). This first approach will be studied in section 3.4.1. AMAS can also be described as using reorganization like in [132] since the organization is clearly modified during the MAS functioning. However, since there is no universally accepted definition of MAS reorganization, in this dissertation we chose to restrict the scope of the term reorganization to top-down approaches which are those that are retained in our proposal. We prefer to retain the term of self-organization to denote the bottom up approaches where, de facto, an adaptation and modification of the emergent organization is installed. Concerning the top-down approach, for which we already described static organizational models in 3.3.2 and outlined the intrinsic limitations, adaptation is not at the level of the agents but at the level of the organization (see Figure 3.7). This reorganization process is thus another way of tackling the possible perturbations of the environment and to realize adaptive systems and will be discussed in sections 3.4.2 and 3.4.3.

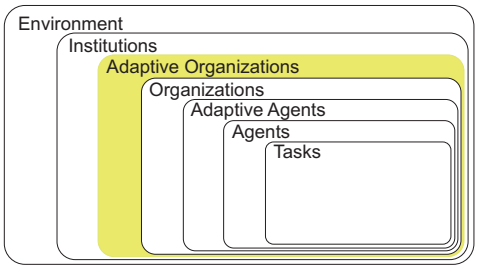


Figure 3.7: Individual to Social View

3.4.1 Self-Organization

As mentioned in 3.2.4, organizations can be either explicit or emergent. In MAS with emergent organizations, called self-organizing MAS, roles are observed a posteriori and no organizational structure is known by the designer.

According to Di Marzo-Serugendo et al. [133], self-organization in artificial systems is the mechanism or the process enabling a system to change its organization without explicit external command during its execution time.

Self-organizing MAS were designed to overcome the disadvantages of explicit organizations when their requirements change due to an unsteady environment. Indeed, MAS which are situated in open and dynamic environments might face severe problems with rigid roles and static organizational structures. For instance, MAS need to manage problems like the variation of the number of agents, changes of task profiles and drop-outs of agents, etc. [134]. Self-organizing MAS must therefore be self-building (able to determine the most appropriate organizational structure for the system by themselves at runtime) and adaptive (able to change this structure as their environment changes).

In order to illustrate this concept of self-organizing MAS, we use the following example taken from [135]. Figure 3.8 represents the interacting agents Pi which constitute the system S.

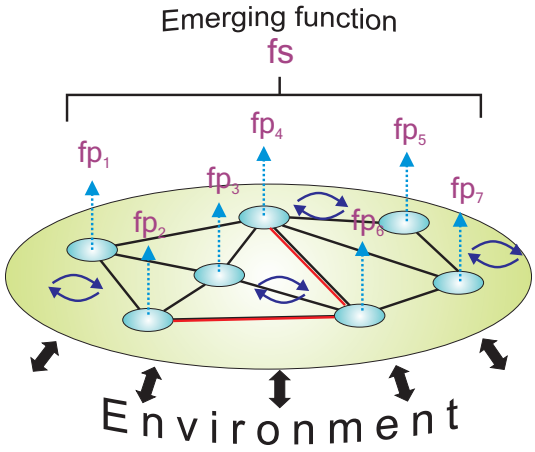


Figure 3.8: Self-Organizational MAS

"We consider that each part Pi of a multi-agent system S achieves a partial function fp_i of the global function fs. fs is the result of the combination of the partial functions fp_i, noted by the operator "**o**". The combination being determined by the current organization of the parts, we can deduce fs=fp_1 **o** fp_2 **o** ... **o** fp_3. As generally fp_1 **o** $fp_2 \neq fp_2$ **o** fp_1, you can change the combination of the partial functions and therefore you modify the global function fs".

Self-organizing MAS have been applied to a wide range of domains. They can take their inspiration from bio-inspired mechanisms, like ant colonies, which were applied for the traveling salesman problem [136] or mobile ad-hoc networks [137], from social and economics approaches like trust-based systems [138] or reputation systems [139] and from fully artificial mechanisms like Self-Organized Public-Key Management for MANETs [140] or self-configuring middleware [141].

One field of application which was of interest in the context of our research was the application to optimization problems. Indeed, complex business optimization problems have been tackled, like in the ATOCA project in which an aircraft design model had to be optimized [142].

This adaptive MAS development methodology using a bottom-up approach allows to deal with distributed optimization. It can adapt to the problem's dynamics and provide solutions at anytime, which are two key issues we want to face in this research. It additionally was applied to business problems. However, since there is no control of the system due to the emergence phenomenon, it is not possible to provide convergence or stability proofs. Although, in the context of our research it is necessary to provide a control on the agents and on their organization since we want to optimize some predefined function(s). We therefore did not further investigated such self-organizing MAS.

3.4.2 Reorganization Dimensions

The organization comes with two aspects, a static one, associated with the organizational structure and a dynamic one, linked to the process that leads to the organization. Attributing a structure to the MAS at the beginning must not imply keeping it at all costs, since an optimal organization is unlikely to be [143]. The MAS has then to be able to adapt himself to the modifications in his environment. Consequently agents have to be able to evaluate the organizational structure in which they live, according to their local goals and their perception of the system and its environment. Moreover, they need special capabilities for modifying this structure. This way, the organization is no more a limit to the autonomy of the agents, neither to the adaptation of the society to the environments' evolutions and contribute to the improvement of the collective performance.

As Hübner says in [125], four dimensions have to be taken into account when dealing with reorganization, What, When, Who and How. Dignum in [144] also proposed a classification of reorganization issues in agents societies. The following draws a parallel between those two classifications.

3.4.2.1 What

According to Hübner, it depends on the organizational model used. The changes can be at the structural level, the functional level, the dialogic level, etc.

Dignum is more restrictive and cites two main situations, a behavioral change (a new agent enters the MAS, an agent leaves the MAS, interaction pattern instantiation) and a structural change (organization self-design and structural adaptation).

3.4.2.2 When

For Hübner, the decision when to start the process can be either static or dynamic. In the first case the process is started according to a predefined criterion fixed within the organizational specification. In the second case, the reorganization process is a consequence of the system functioning, which means that if agents do not meet one or several criteria (goal, performance etc.) the organization is changed.

Dignum defines the ”when” to reorganize as linked to the utility of the organization (interaction success, role success and structure success) and to the utility of the agent (different for each agent, depends on its goals, resource production and consumption).

3.4.2.3 Who

Hübner defines the ”who” takes the initiative as endogenous (one or several agents in the system or auto-organization) or exogenous (MAS user). In the same way, Guessoum in [145] mentions that this observation can be done, either from an external observer (the user of the system himself) or from the system itself. However, she points out that due to their complexity, it appears to be impossible to detect emergent phenomenons using an external observer when dealing with complex systems and thus the system itself has to do it.

Dignum's classification of the ”Who” is based on the C2 model (*Command* and *Control*), i.e. the reorganization decision can be collaborative (consensus) or directive (master-driven). Each of them can be centralized, distributed or even external (outside the organization, e.g. the designer).

3.4.2.4 How

According to Hübner, the reorganization process can be either controlled or emergent. When controlled, rules of reorganization are known in advance. When emergent, an agent will take the decision alone which can lead to the failure of the process (and maybe of the whole system) if it goes against others agents needs.

For Dignum, reorganization is done by *Communications*, which is in fact the third ”C” of the model. They guide decisions on plan and/or social change by sharing information about the environment, the state of the organization, the state of achievement of objectives,etc. Communications then refers to

the meta-observation of the organizational behavior and not to operational communication.

Table 3.4 sums up this parallel between Hübner's and Dignum's reorganization dimensions classification:

	Hübner	**Dignum**
What	depends on the model	Behavioral (agent entrance) and structural changes
When	static (predefined) or dynamic (consequence of the system functioning)	Linked to the utility of the organization (structure success) and of the agent (depends on goals, resources)
Who	endogenous (one or several agents) or exogenous (MAS user)	Collaborative (consensus) or directive (master driven). Centralized, distributed or external
How	controlled (predefined rules) or emergent (single agent's decision)	Communications: meta-observation of organizational behavior

Table 3.4: Hübner's and Dignum's reorganization dimensions classification

3.4.3 Reorganization Approaches

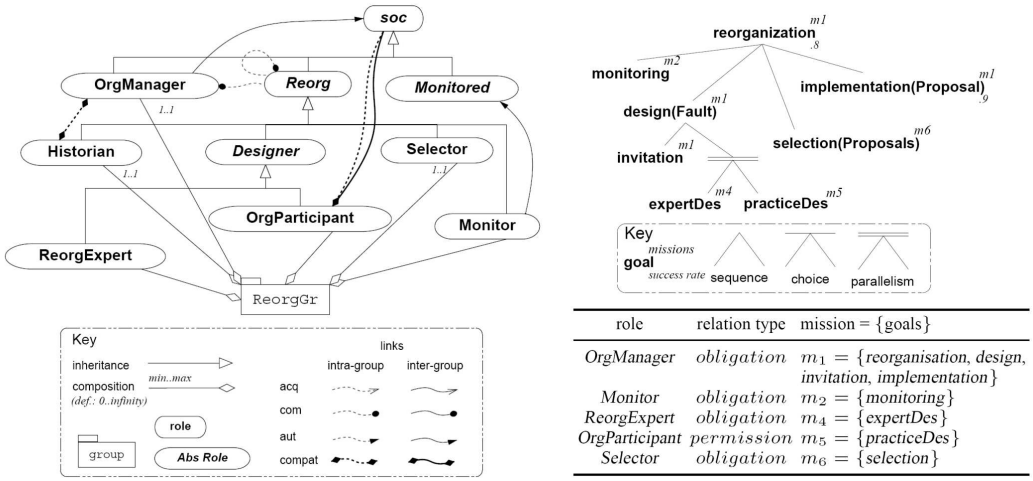
This section provides a survey of existing reorganization approaches in MAS.

3.4.3.1 MOISE+

In MOISE+ (Model of Organization for multI-agent SystEms), Hübner et al. consider the organizational structure and functioning (see Figure 3.9). They concentrate on the controlled changing process which is composed by four phases: monitoring, design, selection, and implementation. In their view reorganization is one cooperative process among others in an MAS and is performed in an endogenous and decentralized way. The reorganization process is performed by a set of agents that play roles inside a group called reorganization group. This group contains a hierarchy of roles that can for example manage, monitor or maintain history of the reorganization process. To summarize, the main contribution is a reorganization model where the agents have autonomy to change their organizations.

3.4.3.2 TAEMS

Horling in [108] uses centralized reorganization process through the TAEMS modeling language and a diagnosis expert subsystem in charge of detecting deficiencies in the organizational model and assisting in the creation of a solution. Its monitoring phase identifies those fails when the system does not behave as expected by its functional model (see Figure 3.10).



role	relation type	mission = {goals}
OrgManager	*obligation*	$m_1 = \{$*reorganisation, design, invitation, implementation*$\}$
Monitor	*obligation*	$m_2 = \{$*monitoring*$\}$
ReorgExpert	*obligation*	$m_4 = \{$*expertDes*$\}$
OrgParticipant	*permission*	$m_5 = \{$*practiceDes*$\}$
Selector	*obligation*	$m_6 = \{$*selection*$\}$

Figure 3.9: Moise+ Reorganization group and Reorganization Scheme

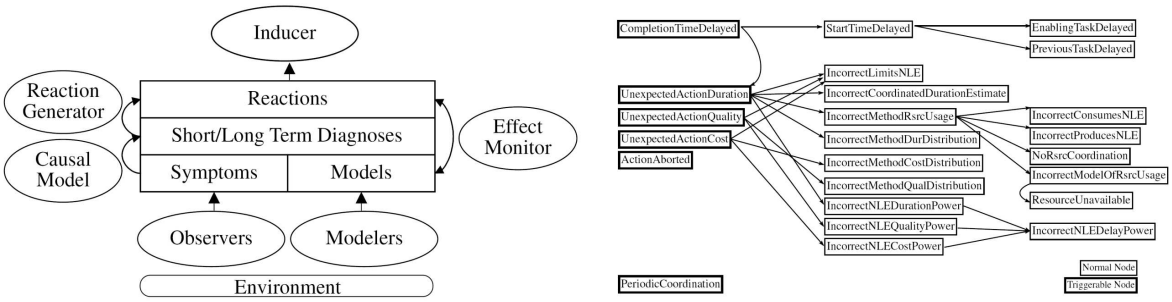


Figure 3.10: TAEMS: high-Level architecture of the diagnostic subsystem and causal model for diagnosing action - and coordination - based faults

3.4.3.3 TEAM

Stone in [146] has a more flexible monitoring phase (see Figure 3.11). His solution is targeted to PTS domains (periodic team synchronization), in which agents act autonomously with limited communication, but they can periodically synchronize in a full-communication setting. Any agent, a soccer player, can identify in the environment the opportunity for reorganization. The reorganization is composed of a change in the team formation and in the current plan (functional level).

3.4.3.4 Chevrier's reorganization model

Chevrier in [147] proposes a model for managing the organization in a MAS restricting it to the notion of interaction management (see Figure 3.12). Interactions are evaluated according to their quality, delay, distance, etc. Based on this evaluation, interactions are qualified in order to choose alternative interactions. The evolution mechanism of the interactions is based on the W-learning algorithm with agents continuously learning so as to react to any perturbation in the interactions during the system's

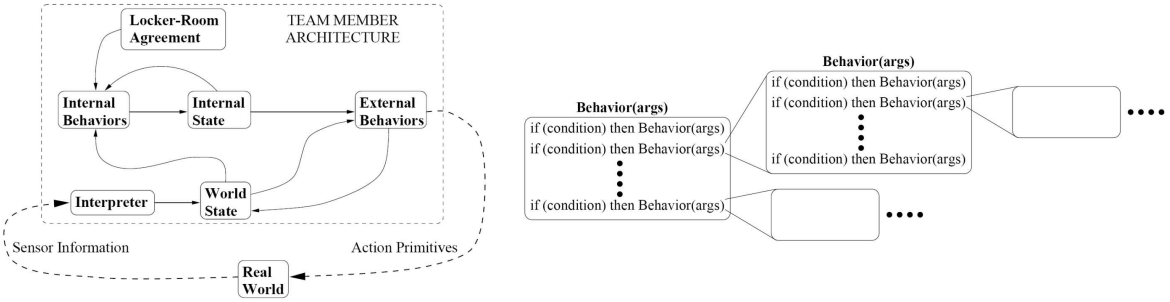
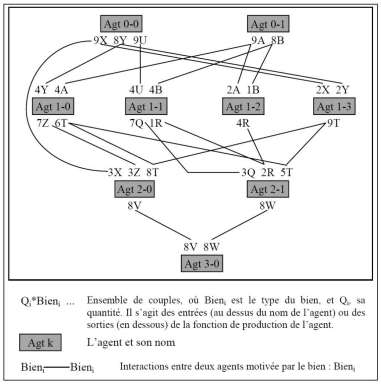


Figure 3.11: TEAM member architecture for PTS domains and internals and externals behaviors organized in an acyclic graph

activity.



Pourquoi ?	Avec qui ?	Comment ?	
		Rôle	Protocole
Obtenir T	Agt 1-0	S	P2.2
Obtenir T	Agt 1-3	S	P2.2
Obtenir R	Agt 1-1	S	P2.2
Obtenir R	Agt 1-2	S	P2.2
Obtenir Q	Agt 1-1	S	P2.2
Fournir W	Agt 3-0	S	P1.2
Obtenir T	Agt 1-0	I	P1.1
Obtenir T	Agt 1-3	I	P1.1
Obtenir R	Agt 1-1	I	P1.1
Obtenir R	Agt 1-2	I	P1.1
Obtenir Q	Agt 1-1	I	P1.1
Fournir W	Agt 1-0	I	P2.1

Figure 3.12: Interaction graph describing an agent society and interaction knowledge base

3.4.3.5 MAGIQUE

Mathieu in [131] proposes in his multi-agent platform MAGIQUE three principles that can be applied to adapt organization: ”have a good address book”, ”share knowledge” and ”recruit new able collaborators” (see Figure 3.13). The number of messages in the multi-agent system is then reduced and the delay before a request is satisfied is improved using the creation of new specific acquaintance relations to remove the middle-agents, the exchange of skills between agents to increase autonomy and the creation of new agents to reduce overloading. Applying those principles implies the modification of the dependance network. Decisions are taken autonomously by the agents and are challenged sometimes to ensure that the chosen acquaintance is still the best choice.

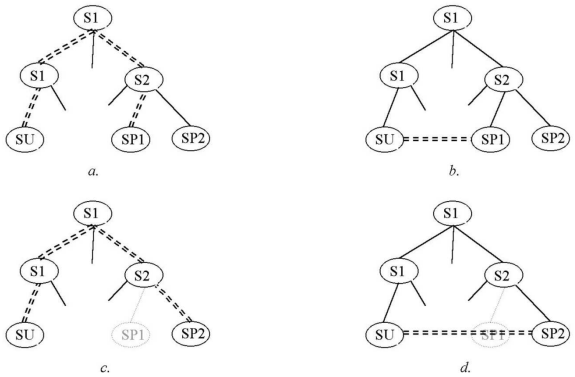


Figure 3.13: Dynamic organization of acquaintances in a multi-agent system.

3.4.3.6 DeLoach's transitional organization model

DeLoach in [148] proposes an organizational model for designing adaptive multi-agent systems. This model possesses three main components: a structural model that contains a set of roles, goals, capabilities and laws, a state model that defines an instance of a team's organization (including a set of agents, the relations between them and the structural model components) and finally a transition function that defines how to switch from one organizational state to another (see Figure 3.14). The concept of transition function is described in detail by Matson in [149], where he also proposes techniques borrowed from model checking as a way to overcome the problem of state explosion when dealing with possible new organizations.

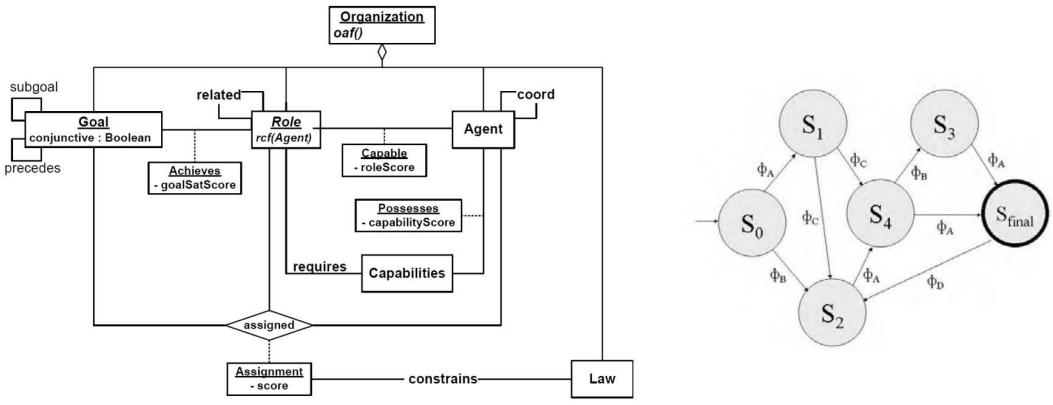
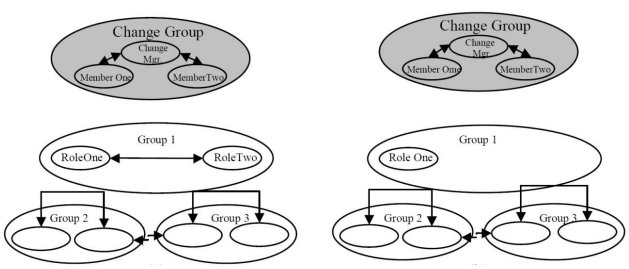


Figure 3.14: Combined structural and state models using standard UML notation and organization transition machine

3.4.3.7 Jonker's organization dynamics formal model

Jonker in [150] introduces a formal modeling approach for organizational dynamics. She uses a formal description language to express the existing and the desired behaviors. The AGR model is used so as to represent the organizational aspects (see Figure 3.15). The structure of the deliberation model is similar to Hübner's with monitoring, goal determination and modification action determination phases. When a change is detected in the organization behavior, a new behavior goal is defined and the structure is modified in consequence. The model is intrinsically complex since there is no predefined reorganization structure, it is fully problem dependent.



RP(Change Manager):Unfreezing Organisation
if the Change Manager has a specification of how the organisation should be changed
and the Change Manager observes that conditions for initiation of the change are met
then the Change Manager unfreezes the organisation by informing the Members about the upcoming organisational change according to the change specification.

RP(Change Manager):Answer Questions
if the Change Manager receives a question from a Member about the upcoming organisational change
then the Change Manager provides an answer to that Member.

RP(Change Manager):Movement to New Organisation Form
if the Change Manager observes that all Member role instances have acknowledged the upcoming organisational change
then the Change Manager announces to the Member role instances when the organisational change will take place.

RP(Change Manager):Refreezing Organisation
if the Change Manager observes that the organisation behaves in a routine manner according to the change specification
then the Change Manager deactivates the Change Group.

Figure 3.15: Organization before and after the change, description of the organizational change properties

3.4.3.8 Ongoing works

As current development we can cite again Horling [151] who, as previously said, is one author of ODML and who mentions in his thesis some first investigations on how to manage organizations' adaptation using ODML. The first step is to monitor problems by observing the deviation of the system compared to the ODML model itself and this will be used as an input for searching for a new configuration. The latter process is no more related to ODML since Horling proposes to either use a set or predefined organization to answer to particular changes or to use various search strategies according to the criticality of the failure. This model is still uncomplete and additionally it is restricted to the case of the knowledge of the full range of possible conditions/modifications in the system.

3.4.4 Synthesis

Table 3.5 describes all the reorganization models previously mentioned according to Hübner's classification (What, When, Who, How).

3.5 Conclusion

One of our contributions lies in using an organizational model to represent coevolutionary genetic algorithms. To this aim, we investigated how MAS can be modeled as organizations.

	What	When	Who	How
Hübner	Depends on the plan of changes written by the designer agents. It can be the whole OS, one group of the SS, one group of the FS or relations of the DS	Dynamic: monitor agents are in charge of checking the validity of the different specifications. If not they refer to the OrgManager	Endogenous: cooperative process performed by a set of agents playing roles in a specific "Reorganization Group"	The Selector Agent chooses on plan of change that will be performed by the OrgManager
Horling	Changes the TAEMS organization that is represented as a goal decomposition tree	Dynamic: the observation is done by the symptoms level in the diagnostic subsystem. It both monitors and builds learning models that can be used as a basis of comparison.	Endogenous: centralized process using an expert subsystem based on a causal model	Controlled: an expert subsystem detects the deficiencies as the system does not behave as expected by the functional model
Stone	Team formation and current plan(functional level)	Dynamic: during team synchronization the team sets globally accessible metrics as formation changing indicators.	Endogenous and decentralized: within the so called "locker-room agreement" every agent takes its reorganization decision(s) during the synchronization	Controlled: team structure includes preplanning for frequent situations (all defined in the locker-room agreement), roles and formations dynamically change based on predefined rules (set in the behaviors)
Chevrier	Modification of interactions	Dynamic: depends on each agent interaction evaluation	Endogenous and decentralized: each agent locally decide to modify its interactions	Agents continuously evolve interactions using a mechanism based on W-learning
Mathieu	Based on MAGIQUE organizational model. It is possible to change the acquaintance organization, the skill distribution and create a pool of apprentices.	Dynamic: each agent takes its own decision according to some predefined criteria (e.g. threshold) or policy.	Endogenous and decentralized: Each agent can take the decision of creating a new communication link, to acquire some skills or to create a pool of apprentices so as to reduce overload.	Controlled: predefined reorganization plan (not explained in detail).
DeLoach	State reorganization (no structure reorganization): modification of roles played by the agents in the team organization.	Dynamic: definition of reorganization triggers: Reorganization for efficiency (accomplishing its goal) and effectiveness (information quality)	Endogenous and centralized: each organization has a global view of the capability function of all its agents	Controlled: organization transition function, computed based on the current state, goals and organizational rules
Jonker	Organizational Structure based on AGR	Dynamic: monitoring agents check in a formal and automated manner the organizational behavior	Endogenous and distributed: the deliberation model (monitoring, goal determination and modification action determination) is embedded in the organization model	Controlled: after monitoring the goal of the organizational behavior is defined and the required modifications a determined

Table 3.5: Reorganization models according to Hübner's classification

In section 3.2, we have seen that there exist several models manipulated within a MAS, i.e. Agent, Environment, Interaction, Organization (AEIO).

As a way to model CGAs, in section 3.3 we first focused on organizations in MAS and on the existing models. Different organizational models have been developed since the end of the nineties and we provided a detailed overview of those we consider as most significant in 3.3.2.

According to our needs, a detailed description of the organization's structure is crucial so as to represent the algorithms topologies and the relations/dependencies between its components. During the first phase of our research we only modeled some static CGAs, therefore, at this time we chose the AGR model. It permitted to simply model the topologies of the algorithms.

However, another issue came during our research: the development of a new dynamic CGA. Consequently, modeling static organizations reaches some limitations which we enlightened in 3.4. We have seen that adaptation in MAS can be based on self-organization (see 3.4.1) or controlled through reorganization (see 3.4.2). A controlled reorganization is necessary to ensure the respect of the CGAs structure, interactions, that is why in 3.4.3 we focused on reorganization approaches which allow to specify the internal dynamics of such systems and/or their interactions with the environment.

We finally chose Moise+ because of the following reasons:
- a richer structural specification compared to AGR which was too restrictive (e.g. no inheritance)
- a functional specification
- an available reorganization extension to the model

The next chapter presents DAFO (Distributed Agent Framework for Optimization), its different models and its implementation. However as it will be discussed in the next chapter, Moise+ still presents some limitations when modeling DAFO's organization. We will therefore introduce MAS4EVO (Multi-Agent Systems for EVolutionary Optimization), a new organizational and reorganizational model based on Moise+ and dedicated to evolutionary optimization.

Part II

DAFO
Distributed Agent Framework for Optimization

Chapter 4

Multi-Agent Model for Coevolutionary Optimization

Contents

4.1 Introduction

In the coevolutionary algorithms literature, the term *Agent* is very redundant when referring to the different players (subpopulations) of these metaheuristics. However, none of the available platforms dedicated to evolutionary computation uses the agent paradigm instead of the object paradigm. Consequently they cannot take benefit from the available agent methodologies and platforms which permit a high level description and an easy development/deployment.

This has motivated the use of the agent paradigm for our DAFO framework dedicated to coevolutionary genetic algorithms. In order to model our framework as a multi-agent organization we introduce MAS4EVO (Multi-Agent System for EVolutionary Optimization), a new multi-agent model dedicated to evolutionary optimization. MAS4EVO and DAFO respectively provide a novel way of modeling and implementing CEAs.

Figure 4.3 provides a general overview of MAS4EVO. In the following we will describe the model structure using the decomposition introduced by the vowel approach AEIO (Agent, Environment, Interaction, Organization) introduced by Demazeau in [86].

The coming chapter is structured as follows. Section 4.2 presents a brief overview on MAS4EVO, section 4.3 introduces the Interaction and Environment models and section 6.2.1 provides a description of the agent architecture used in MAS4EVO as well as details on the different agent types. Section 4.5 describes MAS4EVO's organization model, its different specifications and section 4.6 demonstrates how the CCGA and LCGA are modeled using MAS4EVO. Finally section 4.7 provides our conclusions on the model.

4.2 Model Overview

- As can be seen, MAS4EVO involves different **agents** which can be of three different types, *problem solving agents* in charge of running the optimization process (i.e. the GA, CGAs), *fabric agents* which instantiate the system, and *observation agents* which observe the problem solving agents and provide output interfaces to the end user(s).

- The **environment** represents the optimization problem which is provided by the user. The agents are thus capable of perceiving and modifying their environment so as to evaluate their solutions.

- The **interactions** between the agents are achieved with protocols, either algorithm specific with the exchange of solutions (best, random, etc.) or organization specific with the instantiation and/or modification of the organization (i.e. organizational parameters like topology).

- The **organization** allows to define coevolutionary strategies in terms of their structure, functionalities, communication protocols and norms. It will be detailed in section 4.5.

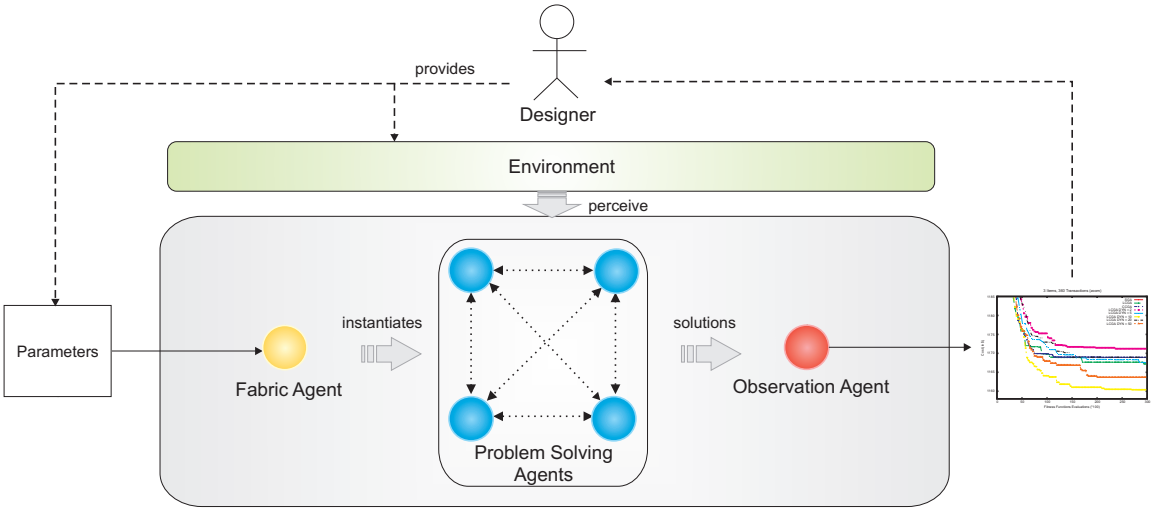


Figure 4.1: MAS4EVO overview

4.3 Interaction and Environment Models

4.3.1 Interaction Model

All the agents in the system must exchange information, concerning either the optimization process or the system's organization. This communication is achieved using interaction protocols with a FIPA-ACL compliant agent communication language. Those communications are composed of propositions between agents, we therefore restricted the set of usable FIPA performatives to *Inform* and *Agree*. For the same reason, we restricted the content expression to FIPA-SL propositions since there is no need for action or identifying reference expression as defined in the FIPA specification [1].

The following introduces the language shared by all the agents of MAS4EVO and the definition of its terms using the EBNF[2] language:

- EvoParameters: parameters set by the user concerning the interaction graph (number of agents, topology of communication) and the algorithm (algorithm type, fitness calculator class, number of experiments, termination condition, termination condition value, number of chromosomes, number of genes, size of genes, crossover type, crossover rate, mutation rate, elitism).

- BestIndividual: the best individual of a genetic algorithm's (sub)population.

[1] FIPA-SL specification: http://www.fipa.org/specs/fipa00008/SC00008I.pdf
[2] Extended Backus-Naur Normal Form

<evoparameters> ::= <interactiongraph> <geneticparameters> <localsearchparameters>?
<interactiongraph> ::= <topology> <numberofagents>
<topology> ::= <string>
<numberofagents> ::= <integer>
<geneticparameters> ::= <algorithm> <fitnessclass> <experiments> <terminationcondition> <terminationconditionvalue> <numchroms> <numgenes> <sizegenes> <crossover> <crossrate> <mutrate> <elitenumber> <localsearch>?
<algorithm> ::= <string>
<fitnessclass> ::= <string>
<experiments> ::= <integer>
<terminationcondition> ::= <string>
<terminationconditionvalue> ::= <integer>
<numchroms> ::= <integer>
<numgenes> ::= <integer>
<sizegenes> ::= <integer>
<crossRate> ::= <double>
<mutrate> ::= <double>
<elitenumber> ::= <integer>
<localsearchparameters> ::= <lsalgorithm> <lsexchangedinformation> <lspopulationrate>? <lsterminationcondition>
<lsalgorithm> ::= <string>
<lsexchangedinformation> ::= <string>
<lspopulationrate> ::= <double>
<lsterminationCondition> ::= <string>

<bestIndividual> ::= <individual>

- BestAndRandomIndividuals: the best and one random individual of a genetic algorithm's (sub)population.

<bestAndRandomIndividuals> ::= <bestIndividual> <randomIndividual>
<bestIndividual> ::= <individual>
<randomIndividual> ::= <individual>

- RandomIndividuals: shuffled list of all the individuals of a genetic algorithm's (sub)population.

<randomIndividuals> ::= <randomIndividual>+
<randomIndividual> ::= <individual>

- PopulationRate: percentage of random individuals of a genetic algorithm's (sub)population.

<populationRate> ::= <individual>+

The four terms previously defined, *BestIndividual*, *BestAndRandomIndividuals*, *RandomIndividuals* and *PopulationRate*, all refer to the term *individual* which consists in a binary chromosome and its assigned fitness value.

<individual> ::= <chromosome> <fitness>
<chromosome> ::= <gene>+
<gene> ::= <allele>+
<allele> ::= "0" | "1"
<fitness> ::= <double>

- Fitnesses: a list of fitness values.

<fitnesses> ::= <fitness>+
<fitness> ::= <real>

- Lifecycle: tells a problem solving agent to start or stop its computation, or informs that a problem solving agent is ready to compute.

<lifcycle> ::= "“Start” | “Stop” | “Ready”

4.3.2 Environment Model

In this multi-agent system dedicated to function optimization, the environment represents the problem to be optimized. This problem can be either static or dynamic and consequently influences the behavior of the system. According to Russel and Norvig [152], the properties of such an environment are:
- Accessible: agents have a perception of the environment. This way agents have the possibility to evaluate their solutions on the optimization problem. The perception of the environment can be either local or global, depending on the agent type and on the algorithm used. However agents do not act on the environment and do not interact through it.
- Non Deterministic: the next state of the environment is not completely determined by the current state of the environment and the actions selected by the agents.
- Episodic: the agents experience is divided into “episodes”. Each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself, because subsequent episodes do not depend on what actions occur in previous episodes.
- Static or Dynamic: depends on the nature of the optimization problem. If the optimization problem is static then the environment is also static. If the optimization problem is dynamic, the environment is also dynamic since they is no synchronization between the agent and the environment (i.e. the environment evolves during the agents' computation).
- Continuous: The perception of the environment can be asked at anytime by the agent (i.e. push-pull mechanism).

There is no dedicated language to describe this environment (i.e. the optimization problem) which is provided by the framework user. The agents can interact with it through one primitive which is:
- Perceive: agents can observe the environment so as to evaluate the fitness of their solutions.

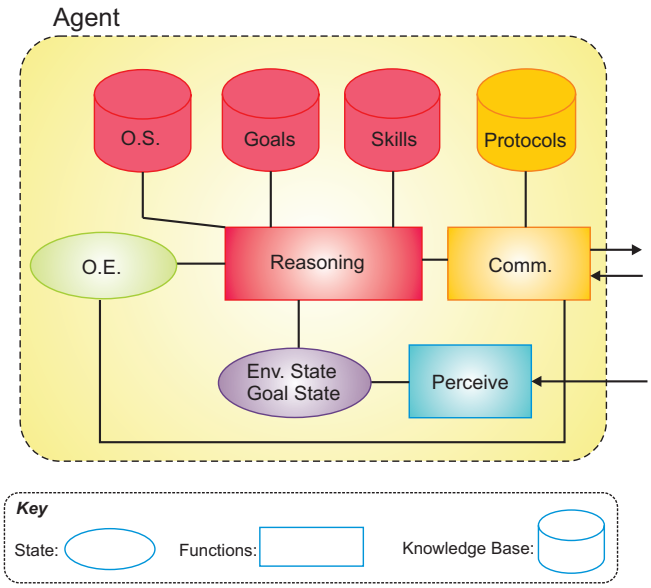


Figure 4.2: Agent structure

4.4 Agent

4.4.1 Global View

DAFO's agents which manipulate goals are therefore built on a cognitive agent model (see Fig. 4.2).

An agent has a set of *goals* it has to achieve, a set of *skills* which allow him to achieve one or several goals. The goals an agent can use are constrained by the *Organizational Entity* (O.E.) in which the agent is situated. The O.E. is an instantiation of one of the available *Organizational Specifications* (O.S.), each O.S. expressing strategies corresponding to a different CGA. An agent additionally knows which goals are satisfied and which are not satisfied through its goals states.

The *perceive* function which permits agents to observe the environment and thus to have information concerning the environment state.

An agent possesses *communication* capabilities with the other agents using a set of interaction *protocols*. As for their goals, the agents' communications are restricted according to the O.E.

The agent architecture comprises an inference engine, referred as *reasoning* function in Fig. 4.2, which selects the goals and skills to execute so as to satisfy the agent's goals, taking into account the O.E.

This architecture is applied on the three types of agents of the system: the *Problem Solving Agents* in charge of the resolution, the *Fabric Agents* in charge of the system management and the *Observation*

Agents in charge of the observation of the Problem Solving Agents and of the interaction with the user.

In the DAFO framework, we can classify the agents in three categories:

- The **Fabric Agents (FA)**: those agents have a global view of the organization and are responsible for the management of the lifecycle of the problem solving agents.
- The **Problem Solving Agents (PSA)**: they are in charge of optimizing a fitness function using a metaheuristic (e.g. a genetic algorithm and/or a local search algorithm). PSA have a partial view of the organization in which they are located.
- The **Observation Agents (OA)**: they have a global view of the problem solving agents and provide interfaces with the user in order to observe the executed computation.

In the following sections we provide details on the three aforementioned agent categories in terms of their functionality, skills and different goals which can be:
- Functional goals: permit to execute specific functions
- Organizational goals: allow agents to modify the organization entity (OE)
- Interactional goals: involve the communication between agents
- Supervisorial goals: dedicated to the monitoring of the organization's state according to some criteria (e.g. in order to trigger a reorganization process).

The following description will provide an exhaustive list of the agents' functionalities, skills and goals, i.e. only a subset of these will be used depending on the instantiated CGA.

4.4.2 Problem solving Agent Model

The *Problem Solving Agent* uses one metaheuristic (a genetic algorithm or a local search algorithm) to optimize a mono-objective function.

4.4.2.1 Goals

Functional Goals:
- gEvaluate(i*): evaluates one or several individual(s) i on a local or global mono-objective fitness function.
- gReEvaluate(i*, f*): re-evaluates all the individuals i of a population according to the fitness value f received from neighbor agent(s) concerning the same individuals.
- gRunGA(op_c**,** p_c**,** p_m**, elite):** executes one generation of a generational GA. The crossover operator op_c is applied with probability p_c and mutation is applied with probability p_m on the population of solutions to obtain a new offspring population to be evaluated. It is possible to add some elitism in order to keep one or several best individual(s) unchanged in the next generation.
- gRunLS(ls_{alg}**, i*,** $ls_{termination}$**):** executes several steps of one local search algorithms ls_{alg} among the five available ones: Steepest Ascent Hill Climbing (SAHC), Next Ascent Hill Climbing (NAHC),

Random Bit Climbing (RBC), Dynamic Hill Climbing (DHC) or Tabu Search (TS). It is applied on one or several individuals $i*$ and stops its search process once the termination condition $ls_{termination}$ is met.

Organizational Goals:
- gCreateGroup(g): creates the group g.
- gCreateSubGroup(sg,g): creates the subgroup *sg* of the group *g*.
- gAdoptRole(r,g): corresponds to the adoption of a role *r* in the group *g*.
- gLeaveRole(r,g): corresponds to leaving the role *r* of the group *g*.

Interactional Goals:
Problem solving agents have the capability of using two interaction protocols (defined in 4.5.4):
- pInform(g, s, r, c): used by the PSA (sender *s* in the group *g*) to send one or several individuals as content *c* to other PSA(s) and to send the best individual to the OA (receivers *r* in the same group *g*).
- pNegociate(g, r, c): used by the PSAs, playing the role *r* in the group *g*, during the reorganization process to negotiate their new groups submitted a content parameter *c*.

Supervisorial Goals:
- gMonitoring(c): monitors of the organization according to a criterion *c* in order to start the reorganization process.

4.4.2.2 Skills

Problem solving agents possess skills in order to fulfill the goals aforementioned. The latter are:
- evaluate: to evaluate a solution on a given optimization problem.
- genGA: to run a generational genetic algorithm (genGA).
- LS: to run a local search algorithm (LS).
- sendToPSA: to send one or several solution(s) to other problem solving agents
- sendSolutionsToObserver: to send solutions to the observation agent.
- createComputingGroups: to create new computing groups and adopting roles in the latter.
- leaveComputingGroups: to leave their computing groups.
- findNewComputingGroups: to negotiate with other problem solving agents so as to find new computing groups.
- joinComputingGroups: to join new computing groups.
- isReoganizationNeeded: to decide when to reorganize (in the case of a dynamic coevolutionary genetic algorithm, dLCGA).

4.4.3 Fabric Agent Model

The *fabric agent* manages the initialization of the system by deploying the problem solving agents within the organization which is described by the user. Moreover in some cases it is able to observe the global computation time and to supervise the problem solving agents computation.

4.4.3.1 Goals

Organizational Goals:
-gInstantiate(a): instantiates the problem solving agents *a* according to the parameters set by the user.

Interactional Goals:
- pInform(g, s, r, c): used by the FA (sender *s* in the group *g*) to send the parameters set by the user as content *c* to all the other agents (PSA and OA) as receivers *r* in the same group *g*.
-pTime(g, s, r, c1, c2, c3, n): used by the FA (sender *s* in the group *g*) to send messages (with content *c1*, *c2* and *c3*) to all the other agents (PSA and OA) as receivers *r* in the same group *g* according to some time interval *n* in order to ensure the respect of some computational time constraint.

4.4.3.2 Skills

The fabric agent possesses skills in order to fulfill the goals aforementioned. The latter are:
- parseEvoParameters: to parse the parameters set by the user.
- createPSAs: to create problem solving agents according to the user information.
- sendParametersToPSA: to send parameters to the problem solving agents.
- sendParametersToObservation: to send parameters to the observation agent.
- isComputationalTimeFinished: to observe the global computational time.
- sendStartToPSA: to send a start message to the problem solving agents in order to start a computation with a termination condition based on a number of generations of fitness functions evaluations.
- sendStopToPSA: to send a stop message to the problem solving agents in order to respect a computational time termination condition.

4.4.4 Observation Agent Model

The *observation agent* monitors the *problem solving agents*, i.e. the results they provide and saves this information in log files and optionally in graphics.

Functional Goals:
- gEvaluate(i*): evaluates the individuals i on a global mono-objective fitness function.
-gLog: saves the computed best fitness of all generations of a run of the algorithm in a file. It also

saves the average fitness per generation over all the runs in a separate file.
-gGraphandLog: similar to gLog but additionally draws the corresponding fitness graphs (current run and average) during the system's functioning.

4.4.4.1 Skills

The observation agent possesses skills in order to fulfill the goals aforementioned. The latter are:
- aggregateReceivedSolutions: to aggregate the local best solutions of each generation of the problem solving agents.
- evaluateGlobalSolution: to compute the fitness of the global solution on the global mono-objective fitness function.
- logBestInGeneration: to create a log file for each run of the algorithm containing the computed best fitness per generation.
- computeAverage: to compute the averaged best fitness per generation over all the algorithm runs.
- logAveragedBestInGeneration: to create a log file to save the average best fitness of all the generations.
- drawBestInGeneration: to draw a graph of the best fitness per generation of the current run.
- drawAveragedBestInGeneration: to draw a graph of the averaged best fitness per generation over the runs.

4.5 Organization Model

Within the MAS4EVO model, we proposed a new organizational model based on Moise+ and dedicated to evolutionary optimization. This model allows to structure the functioning of the DAFO framework and to constrain the behavior of each agent participating in. Through the use of the different specifications of MAS4EVO, it is possible to model different evolutionary algorithms by constraining similar agents in terms of their functioning and behavior.

4.5.1 Overview

The MAS4EVO model is based on Moise+ and extends it. Moise+ is an organization centered model which considers three different specifications. Indeed it allows to specify the structure of an agents' organization in terms of roles, groups and links (structural specification), and the global organization functioning (functional specification). Moise+ adds a deontic relation among these first two dimensions (deontic specification) to better explain how a MASs organization collaborates for the social purpose and makes the agents able to reason on the fulfilment of their obligations or not.

The first limitation encountered when modeling CGAs concerns the specification of the interactions between the problem solving agents (i.e. the subpopulations). For instance, in LCGA it is a two-way communication (a PSA sends random individuals to its neighbor(s) and then receives the

corresponding fitness values) while in CCGA it is a one way communication in which a different type of information is exchanged (i.e. best or best and random individuals). It is consequently necessary to specify these interactions. We therefore added the dialogic specification (DiS) which allows to specify parameterizable generic interaction protocols involving some roles in one or several groups. According to our needs, we also consider that expressing only functional goals in the functional specification is too restrictive. By adding organizational goals in this specification, it is possible to express the actions that an agent playing a role can take on the organization (e.g. to adopt a new role, to create other agents, etc.). Using supervisorial goals, it is possible to monitor the organization so as to specify when to start a reorganization process. We also added interactional goals, which allow to specify when the different generic interaction protocols defined in the dialogic specification are used. Finally, as Moise-Inst extended Moise+'s deontic specification into a normative specification, we extend the deontic expressions into a set of norms. It is thus possible to link a role played by an agent to the different specifications by normative expressions.

Our organizational model for evolutionary optimization is thus composed of:
- A Structural Specification (SS) defining the roles played by the agents, the relations between these roles and the groups to which the roles belong to.
- A Functional Specification (FS) defining the goals which must be achieved by the organization. These goals may concern functional goals, organizational goals, interactional goals and supervisorial goals.
- A Dialogic Specification (DiS) defining a set of generic interaction protocols which can be used by different roles to achieve interactional goals.
- A Normative Specification (NS) defining the right and duties of each role or group .

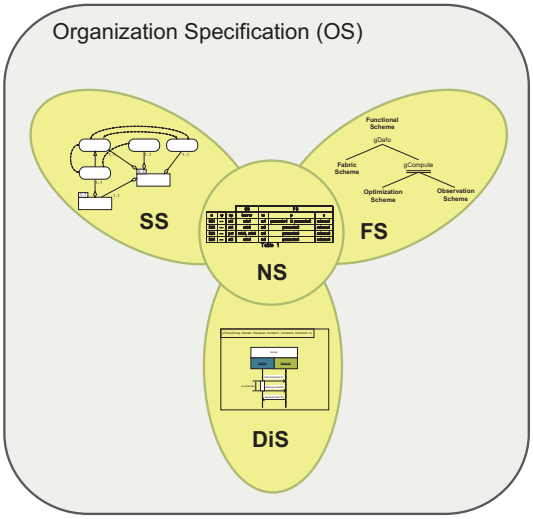


Figure 4.3: MAS4EVO overview

These four specifications form the Organizational Specification (OS) as presented in Figure 4.3 (representation taken from [24]). When a set of agents adopts an OS they form an Organizational Entity (OE). Once created, the history of the O.E. starts and runs by events such as agents entering and/or leaving the OE, group creation, role adoption, mission commitment, etc.

The following sections provide a detailed description of those four specifications. The example of the usage of this model to represent a simple GA within MAS4EVO will be used along these descriptions in order to illustrate the different concepts those specifications are based on.

4.5.2 Structural Specification

The Structural Specification (SS) expresses the structure in terms of *roles*, *relations* between roles and *groups*. A set of constraints expresses the links scope and the cardinality of roles and groups. The SS is graphically represented using a formalism as shown in Figure 4.4.

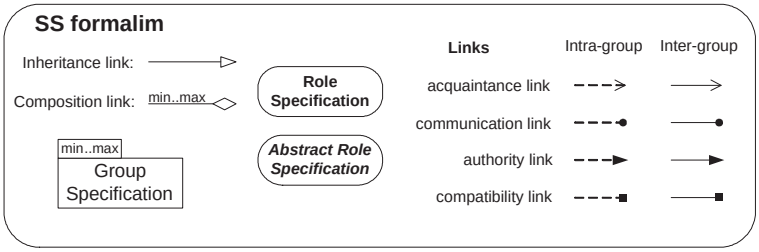


Figure 4.4: Structural Specification formalism

Roles: a *role* is a label which allows to define a set of constraints an agent has to respect as soon as it accepts to play that role. A role is also used to define structural constraints with other roles and groups. Finally, a role is a label which is attached in norms (see 4.5.5). For specification purposes it is possible to define *abstract roles* but they cannot be played by any agent. It is also possible for one role to inherit the properties from a parent role using an *inheritance relation* (an abstract role can not inherit from a non-abstract role). If one role r' inherits from a role *r* and *r* is different from *r'*, then *r'* receives the properties of *r* and *r'* becomes a sub-role of *r*. A role can not inherit from himself and can not inherit from an abstract role.

Groups: a group is defined by a composition of non-abstract roles (composition link), a set of intra-group links, a set of inter-group links , roles and groups cardinalities and agent cardinality (i.e. the number of agents which can play a role in the group). Any non-abstract role is obliged to belong to a group while abstract roles do not belong to any group. An inheritance link connects two roles together.

Links: links are the relations which have a direct influence on the agents' behavior (contrary to the inheritance). They can be of four types: acquaintance (*acq*), communication (*com*), authority (*aut*) and compatibility (*comp*).
- An *acq* link means that an agent playing a source role is authorized to have a representation of the agents playing the destination role.
- In a *com* link the agents playing the source role can communicate with the agents playing the destination role. A communication link implies the existence of an acquaintance link ($com \rightarrow acq$).
- In an *aut* link the agents playing the source role can control the agents playing the destination role. An authority link implies the existence of a communication link that implies the existence of an acquaintance link. An authority link implies the existence of a communication link and thus of an acquaintance link($aut \rightarrow com \rightarrow acq$).
- In a compatibility link agents playing the source role are authorized to play the destination role. By default, the roles of a SS are not compatible.

Links can have two different scopes: intra-group and inter-group. An intra-group link sets the scope of the link to the agents playing the source role to the agents playing the destination role in the same instance of a group or its sub-group. An inter-group link sets the scope of the link to the agents playing the source role to the agents playing the destination role independently of the instances of the group.

Figure 4.5 shows the links usage in the SGA example. The *EvoBuilder* role has an intra-group authority link on the *EvoMember* role since the *EvoBuilder* is in charge of instantiating the *EvoMember* roles. The *EvoMember* role has an intra-group communication link with the *EvoBuilder* role since it will send information on its status. The *Solver* has an intra-group communication link with the *Observer* since the *Solver* will send information concerning its computation. Both roles do not directly belong to the same group, but since the *SGA* group is a subgroup of the *DAFO* group the link is intra-group and not inter-group. Finally the *EvoMember* role and the *Solver* role have an intra-group compatibility link which means that the same agent can play both roles in the same instance of the *DAFO* group (we do not mention the *SGA* group since it is a subgroup of the *DAFO* group).

Cardinalities: cardinalities constraints specify the minimum and maximum number of roles or group instances that a group instance accepts. Three types of cardinalities can be used within a group: role, sub-groups and agent cardinalities.
- *Role cardinality*: specifies the minimum and maximum number of role instances accepted by the group instance, i.e. the number of agents playing that role.
- *Sub-group cardinality*: specifies the minimum and maximum number of instances of a sub-group that is accepted by the parent group instance.
- *Agent cardinality*: introduced in Moise-Inst, this cardinality specifies the minimum and maximum number of agents capable of playing a role in the instance of a group and its sub-groups. This is of

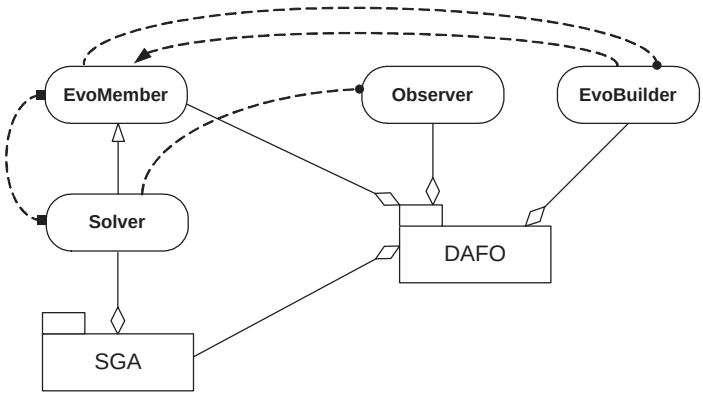


Figure 4.5: Example of links usage (Structural Specification)

interest when compatible roles are in the same group.

Figure 4.6 shows the cardinalities usage in the SGA example. The *EvoMember*, *Observer* and *EvoBuilder* roles have role cardinalities equal to (1..1) which means that only one instance of each role will be accepted by the *DAFO* group instance. The *SGA* group has a sub-group cardinality of 1..1 with the *DAFO* group, which means that a single instance of the *SGA* group can be contained in one instance of the *DAFO* group. The *SGA* group has an agent cardinality of 1..1 which means that a single agent can play the Solver role in it. The same way the *DAFO* group has an agent cardinality of 3..3 which means that only three agents will play the four different roles contained in this group (*EvoMember*, *Observer*, *EvoBuilder*, *Solver*). One agent will therefore play two different roles.

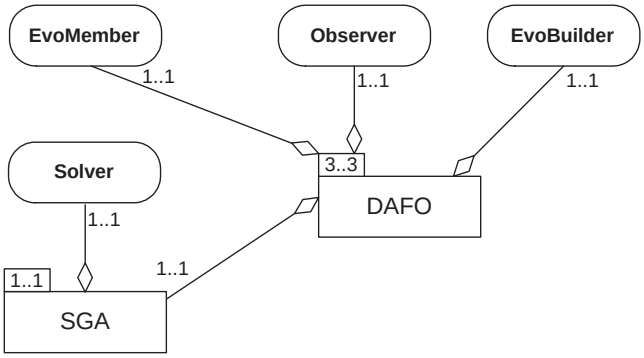


Figure 4.6: Example of cardinalities usage (Structural Specification)

4.5.2.1 Roles

As already mentioned, one advantage of MAS4EVO is its capacity to model different algorithms by using similar agents with different organizations. This section provides an exhaustive list of the roles

that can be played by the agents of MAS4EVO:

- Observer: role which can be played by the *Observation Agent*. This role is in charge of both evaluating the global fitness in each generation of the algorithm and logging/drawing the graph of the obtained fitness values, based on the information received from the *Problem Solving Agent(s)*.

- EvoBuilder: role which can be played by the *Fabric Agent*. This role manages the organization through the instantiation of the *Evolutionary Agents* , the initialization of the *Evolutionary Agents* and the *Observer Agent*.

- EvoMember: role which can be played by *Problem Solving Agent(s)*. Roles which skills include running optimization algorithms will inherit from this *EvoMember* role.

- Solver: role which can be played by the *Problem Solving Agent(s)*. This role can communicate individuals (best and/or random, population ratio, etc.) with the agents playing the *Observer* role, the *LocalSearcher* role and other agents playing the *Solver* role.

- Producer: role which can be played by the *Problem Solving Agent(s)*. It is in charge of communicating individuals (best and/or random) to the *Consumer(s)*.

- Consumer: role which can be played by the *Problem Solving Agent(s)*. It is in charge of running the SGA based on the individuals coming from the *Producer(s)*.

- LocalSearcher: role played by the *Problem Solving Agent(s)*. It runs one of the available local search algorithms on the individual received from the *Producer* located in the same group and sends back the possibly improved individual to the *Producer* after a predefined termination condition (i.e. a number of function evaluations).

4.5.2.2 Groups

Similarly to the roles, this section provides an exhaustive list of the groups in which agents can play the roles listed in the previous section (4.5.2.1) in the DAFO framework:

- DAFO: base group which contains all the agents and groups of the organization. The roles played in this group are *Observer*, *EvoBuilder* and *EvoMember*.

- Solving Unit: group which contains exactly one *Consumer* and one *Producer*, the two roles being played by two different *Problem Solving Agents*.

- LCGA: this group contains all the *Solver(s)*, *Consumer(s)* and *Producer(s)* representing the LCGA, dLCGA or the coevolutionary part of the hLCGA.

- CCGA: this group contains all the *Solver(s)*, *Consumer(s)* and *Producer(s)* representing the CCGA.

- LocalSearch Unit: this group contains exactly one *Producer* and one *LSAgent* being played by two different *Problem Solving Agents* and representing the hybridization of the LCGA.

4.5.2.3 Links

The following table lists all the possible links between all the possible roles in the DAFO organization.

		Destination Role						
		Observer	EvoBuilder	EvoMember	Solver	Producer	Consumer	LocalSearcher
Source Role	Observer	-	-	-	-	-	-	-
	EvoBuilder	-	-	aut	-	-	-	-
	EvoMember	-	com	-	-	-	-	-
	Solver	com	-	-	com	-	-	com
	Producer	-	-	-	-	-	com	-
	Consumer	-	-	-	-	com	-	-
	LocalSearcher	-	-	-	com	-	-	-

Table 4.1: Possible links between roles

In order to model different CGAs, the DAFO organization modeled using MAS4EVO brings constraints on the possible links between the different available agents. For instance, for the *Solver* role, it has a communication link with the *Observer* role, it can have a communication link with one or several other *Solver* roles (depending on the CGA) and a communication link with a *LocalSearcher* in case of a hybrid algorithm (see 5.2).

4.5.2.4 Structural Specification Example

This section provides the example of the structural specification of a simple GA within DAFO using MAS4EVO. This example will be used to illustrate the four different specifications of MAS4EVO.

The root group is the *DAFO* group which is composed of a single group *SGA* (cardinality "1..1"). The *DAFO* group is composed of three agents (cardinality "3..3"), one playing the *EvoBuilder* role, another one the *Observer* role and another one the *EvoMember* role. Indeed each of these roles can only be adopted once (cardinality "1..1"). The *SGA* group is composed of one agent (cardinality "1..1") which will play the role *Solver* due to its "1..1" cardinality. The *Solver* role inherits from the *EvoMember* role.

The *EvoBuilder* role possesses an authority link on the *EvoMember* role since it will control the lifecycle of the *EvoMember* (e.g. instantiation). The *EvoMember* possesses a communication link with

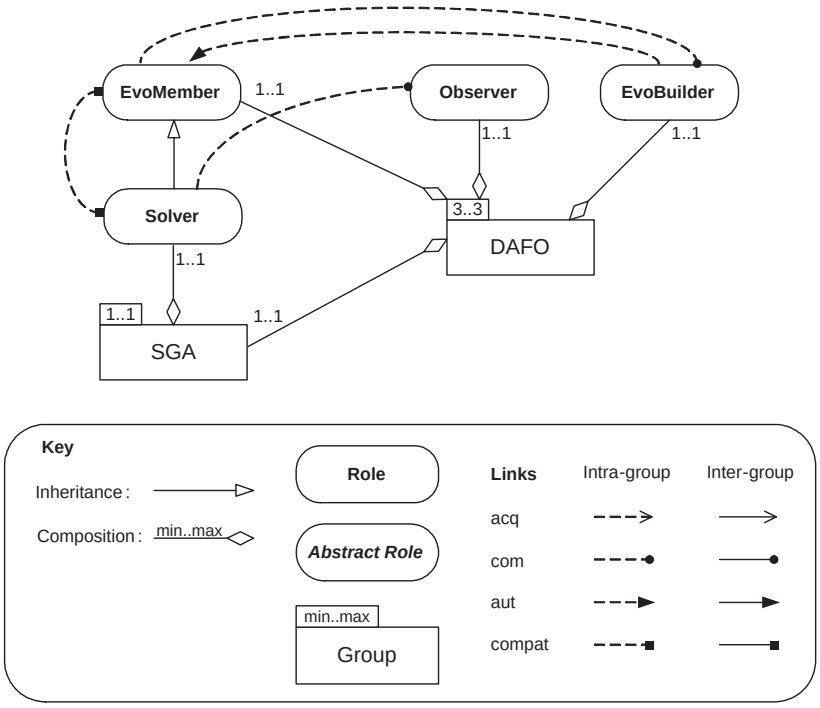


Figure 4.7: SGA Structural Specification

the *EvoBuilder* role which will be used to provide information concerning its lifecycle (e.g. ready to compute). The *Solver* role also has a communication link but with the *Observer* role so as to send results of its computation process (e.g. the best individual per generation). Finally the *EvoMember* role and the *Solver* role have a compatibility link which means that the same agent can play both roles in the same instance of the *DAFO* group (we do not mention the *SGA* group since it is a subgroup of the *DAFO* group).

4.5.3 Functional Specification

The Functional Specification (FS) describes a set of social schemes considered as the global (collective) goals to be achieved by the organization. These goals are grouped into missions which will be distributed to the agents. MAS4EVO extends this specification by enlarging the scope of goals which are not only functional ones as in Moise+ but also interactional, organizational or supervisorial goals. MAS4EVO additionally provides the possibility to associate a repetition constraint on a goal.

A social scheme can be seen as a goal decomposition tree where the root is a global goal and the leaves are goals that can be achieved by one agent. Missions group the goals in a coherent set which will have to be accomplished by the agents. The graphical representation of the functional specification uses a formalism as shown in Figure 4.8.

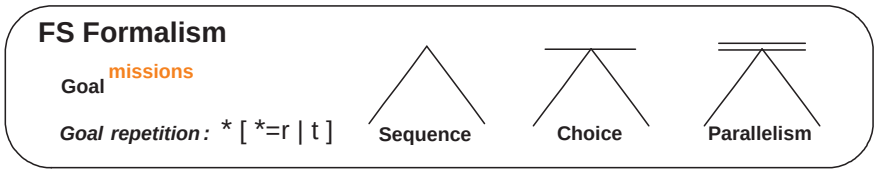


Figure 4.8: Functional Specification formalism

Social Scheme: a *social scheme* is a structured plan allowing to achieve a global goal of the organization. A social scheme is decomposed in goals structured in plans and grouped in missions. The main *social scheme* of a FS composed of several *social schemes* is called *functional scheme*

Goals: a *goal* represents a state to be reached by the organization. MAS4EVO allows to specify different types of goals which can be:
- functional goals: as used in Moise+.
- interactional goals: use one generic interaction protocol specified in the dialogic specification. It must mention the source and destination roles as well as the information exchanged if necessary.
- organizational goals: imply an action on the organization entity (e.g. instantiation of new agents). Therefore they provide reorganization capabilities.
- supervisorial goals: imply a monitoring of the organization so as to start a plan once a predefined criterion is reached. They can be used to trigger a reorganization process.
- artificial goals: goals that can not be reached, they are used only for specification purpose.
A goal can be repeated for number of iterations or until a predefined termination condition is met (e.g. a time constraint). This repetition constraint is expressed in the graphical representation of the FS as follows: *[*=n | t] with n representing the number of iterations, t representing a termination condition and | being an OR operator.

A *plan* achieving a goal of a social scheme decomposes this goal in sub-goals using three possible operators:
- sequence ",": the plan "$g_1 = g_2, g_3$" means that the goal g_1 will be achieved if the goal g_2 is achieved and after that also the goal g_3 is achieved.
- choice "|": the plan "$g_1 = g_2 \mid g_3$" means that the goal g_1 will be achieved if one, and only one, of the goals g_2 or g_3 is achieved.
- parallelism "‖": the plan "$g_1 = g_2 \parallel g_3$" means that the goal g_1 will be achieved if both g_2 and g_3 are achieved. Those two goals (g_2 and g_3) can be achieved in parallel.

As introduced in Moise-Inst, we add the possibility to make a reference to a social scheme inside a plan. This means that for instance if the plan of the goal g_1 is a choice between a goal g_2 or a scheme sch_1, then g_1 will be accomplished if either the goal g_2 is achieved or the root goal of sch_1 is achieved. It is of interest when the same set of goals is usable several times in the same FS (it avoids

to mention the same goals multiple times). It is also possible to use the same goal in different social schemes which is a way to synchronize those schemes.

Compared to Moise+, MAS4EVO provides additional information concerning the satisfaction of a parent goal if it is part of mission(s) or not. We thus consider that to satisfy a parent goal:
- if no mission is attached to the parent goal, all its sub-goals must be satisfied. This is illustrated on the left hand side of Figure 4.9, in which the parent goal gOptimize is satisfied once gRunGA and pInform(best) are satisfied.
- if one or several missions are attached to the goal, all its sub-goals must be satisfied and the agents engaged on these missions must be satisfied. This is illustrated on the right hand side of Figure 4.9, in which the parent goal gOptimize is satisfied once gRunGA and pInform(best) are satisfied and the agents engaged on the goal gOptimize agree that it is satisfied.

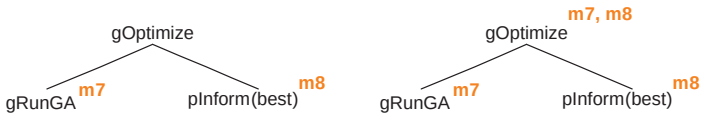


Figure 4.9: Example of missions usage (Functional Specification)

Mission: a *mission* is a set of coherent goals belonging to the same scheme that a role can commit to. An exception exists in the case of an interaction goal, where a mission is attached to two or more roles. Logically, an artificial goal can not be part of a mission since no role can satisfy it.
For each mission a cardinality is defined specifying the number of roles which can be involved in the mission at the same time. A cardinality is expressed in the following form:
- n..m: with $n \in \mathbb{N}$, $m \in \mathbb{N}$ and $n \geq m$ (example: 1..1 or 2..4)
- n..*: equivalent to n or more with $n \in \mathbb{N}$ (example: 6..*)
By default the cardinality is (0..*).

4.5.3.1 Functional Specification Example

Figure 4.10 represents the functional specification of the SGA within MAS4EVO. The specification is built on four social schemes, a functional scheme (considered as the main scheme of the FS which root goal *gDafo* is to run the DAFO framework), a fabric scheme (which root goal *gOrganize* is to initialize the problem solving agent), an optimization scheme (which root goal *gOptimize* is to optimize a problem) and an observation scheme (which root goal *gOutput* is to provide an output of the results obtained by the GA). All these root goals are artificial goals and consequently are not attached to a mission as specified in 4.5.3.

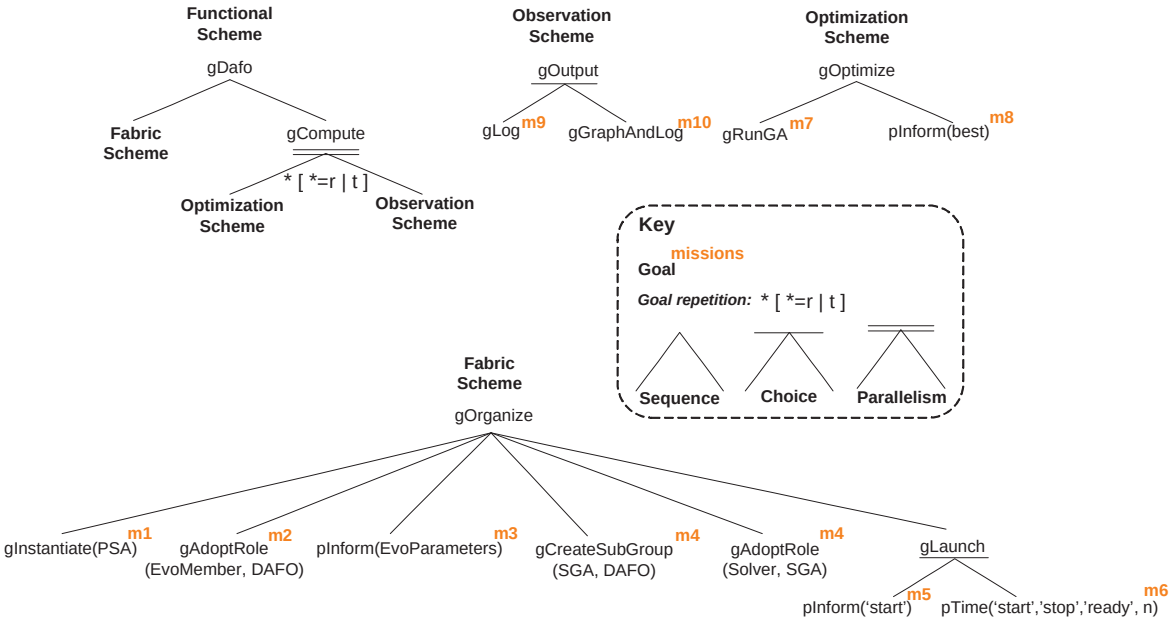


Figure 4.10: SGA Functional Specification

The goal *gDafo*, root goal of the functional scheme, is satisfied when the *Fabric Scheme* and the artificial goal *gCompute* are sequentially satisfied. The goal *gCompute* is itself satisfied once repeated for a number n of iterations or repeated until a time termination condition of t seconds is met. An iteration of the *gCompute* goal will be achieved when the *Optimization Scheme* and the *Observation Scheme* are realized in parallel.

The artificial root goal of the *Fabric Scheme* (*gOrganize*) is satisfied when the following goals are sequentially satisfied:
- *gInstantiate(PSA)* organizational goal instantiating the Problem Solving Agent
- *gAdoptRole(DAFO, EvoMember)* organizational goal in which the *EvoMember* role is taken in the *DAFO* group
- *pInform(EvoParameters)* interactional goal in which the parameters set by the user are transmitted
- *gCreateSubGroup(DAFO, SGA)* organizational goal in which a *SGA* group is created as a subgroup of *DAFO*
- *gAdoptRole(SGA, Solver)* organizational goal in which the *Solver* role is adopted in the *SGA* group
- *gLaunch* artificial goal satisfied when either the interactional goal *pInform(start)* or the interactional goal *pTime(start, stop, ready, n)* are satisfied. *pInform(start)*, in which a "start" message is sent, will be satisfied in case of a termination condition based on a number of generation or a number of fitness function evaluations. *pTime(start, stop, ready, n)*, in which start and stop messages are sent with a time interval, will be satisfied in case of a time termination condition.

Concerning the artificial root goal *gOptimize* of the *Optimization Scheme*, it is satisfied once the functional goal *gRunGA* running one generation of a SGA and the interactional goal *pInform(best)* in which a message containing the best individual are sequentially satisfied.

Finally the artificial root goal *gOutput* of the *Observation Scheme* is satisfied if either the functional goal *gLog* saving the received best individual of one generation and the calculated average in each generation in log files is fulfilled or if the functional goal *gGraphAndLog* saving the received best individual of one generation and the calculated average in each generation in log files and drawing the corresponding graphs is fulfilled.

Id.	Goals of the mission	Card.	Description
m1	gInstantiate(PSA)	1..1	Instantiate the Problem Solving Agents
m2	gAdoptRole(EvoMember)	1..1	Adopts the EvoMember role in the EvoFramewok group
m3	pInform(EvoParameters)	3..3	Sends the Evoparameters
m4	gCreateSubGroup(DAFO, SGA), gAdoptRole(SGA, Solver)	1..1	Creates the subgroup SGA and adopts the Solver role in it
m5	pInform(start)	2..2	Sends a start message
m6	pTime(start,stop, ready, n)	2..2	Sends start and stop messages with a time interval n
m7	gRunGA	1..1	Runs a generation of genetic algorithm
m8	pInform(best)	2..2	Sends the best indiv. of a generation of the SGA
m9	gLog	1..1	Logs the best and average fitness
m10	gGraphAndLog	1..1	Logs the best and average fitness and print the corresponding graphs

Table 4.2: Missions definition of the FS of the SGA

The missions we define for the SGA example are reported in Table 4.2. The Fabric Scheme groups the missions *m1*, *m2*, *m3*, *m4*, *m5* and *m6* concerning the instantiation of the OE and the startup of the computation process. *m1*, *m2* and *m4* have a cardinality (1..1) since respectively only one fabric agent and one problem solving agent can be involved in these missions. *m2*, *m5* and *m6* have a cardinality (2..2) since they include interactional goals which require at least two agents.

The same way, the optimization scheme groups the missions related to the GA computation, the functional goal *gRunGA* is set in a mission (*m7*) to run a generation of a GA and the interaction goal *pInform(best)* is set in a mission (*m8*) to communicate the best individual found after this computation.

Finally, the Observation scheme groups the missions *m9* and *m10* related to the observation of the computation, which either log the best fitness received and the calculated average fitness or log the best fitness received, the calculated average fitness and prints the corresponding graphs.

4.5.4 Dialogic Specification

The main concepts of this specification are issued from AUML sequence diagrams [22] and from AGR organizational sequence diagram introduced by Ferber in [23] of which we use the notion of groups

and roles.

The Dialogic Specification (DiS) allows to specify parameterizable generic interaction protocols independent from the roles and groups specified in the structural specification (SS). These interaction protocols are a variant of both AUML sequence diagram [22] and AGR organizational sequence diagram [23].

The graphical representation of the DiS has two dimensions: 1) the vertical dimension represents the time ordering and 2) the horizontal dimension represents generic groups and roles that will be specified as parameters. Messages in sequence diagrams are ordered according to a time axis. This time axis is usually not rendered on diagrams but it goes according to the vertical dimension from top to bottom. Message ordering is expressed by the time axis. The graphical representation of the dialogic specification uses a formalism as presented in Figure 4.11.

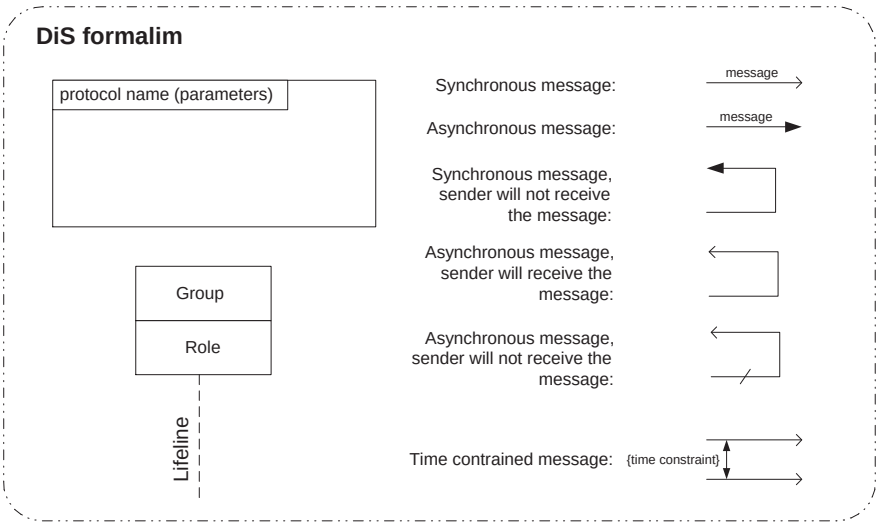


Figure 4.11: Dialogic Specification formalism

4.5.4.1 Interaction:

As for UML and AUML, an interaction protocol is encapsulated in a frame rendered as a solid-outlined rectangle. In the upper-left corner of the frame a smaller rectangle contains the name of the protocols and the parameters linked to it. The parameters refer to the group(s), role(s) and also to the content (which is not specified in protocol templates in AUML) that correspond to the instantiation of one generic protocol.
Protocol designers also need to know which ontologies, content language and agent communication language are used in this protocol.

Constraints:
- Protocol names must be unique in the DiS.
- To instantiate a protocol, the number of parameters specified in its definition must be respected.

4.5.4.2 Lifelines

Contrary to the specification of UML 2.0 and restricting the possibilities of AUML, a lifeline necessarily represents a role played by one or more agents in a group. The lifeline in sequence diagrams defines the time period during which a role exists for this interaction, represented by vertical dashed lines. When a lifeline is created for a role, this role becomes active for the protocol. This lifeline is present as long as the role remains active in the protocol.
A lifeline is composed of three elements: two labels depicted in two different boxes, one above the other, on the top of the lifeline and a vertical dashed line anchoring to the box. The upper label is used for the group and the lower label for the role played in this group.
It is preferable to order lifelines according to the appearance of the roles in the interaction. As a lifeline corresponds to the appearance of the role in the interaction, it is then important to shift the lifeline to the bottom if this role appears late in the interaction.

Constraints:
- A pair (role, group) must appear one and only one time in the sequence diagram.

4.5.4.3 Messages

As in UML and Agent UML, a message defines a particular communication between two lifelines in the sequence diagrams. Senders and receivers of a message can be on the same lifeline or not. Two situations have to be considered when the sender lifeline is the same than the receiver lifeline: 1) the sender wants to receive the message as well, 2) the sender wants to be omitted from the set of recipients. These two different situations will be distinguished on the message notation.

Agents can use either a synchronous or an asynchronous communication. An asynchronous message means that agents send the message without yielding control. A synchronous message means that agents send the message with yielding of the thread of control (wait semantics), i.e. the agent role responsible of this synchronous message sending waits until an answer message is received and nothing else can be processed. The agent other roles continue to send and receive messages for this protocol or other protocols.

Notation:
A message is shown as a directed line from the sender role to the receiver role. Asynchronous messages have a filled arrow head, synchronous messages have an open arrow head. When a message is sent and received by the same lifeline, it is necessary to consider if the sender will receive the message as well. The directed line is barred near the beginning of the line if the sender does not want to receive

the message. Barring or not the line is only for asynchronous messages. For synchronous messages, the line is always barred to prevent deadlock.

Constraints:
- The content of the message must be specified above the arrow.

4.5.4.4 Timing Constraint

Time on sequence diagrams allows designers to represent that some messages have to be sent after a certain delay. In MAS4EVO timing constraints are relative, i.e. they refer to a specific event in the interaction, the last message for instance. The message to which the timing constraint is applied has to be sent exactly after the specified amount of time represented in curly brackets like 10 s.

This time constraint is rendered as a horizontal bar on the first message, a horizontal bar on the constrained message, a vertical line directed in both ways between the two bars and the timing constraints *time* near the vertical directed line.

4.5.4.5 Dialogic Specification Example

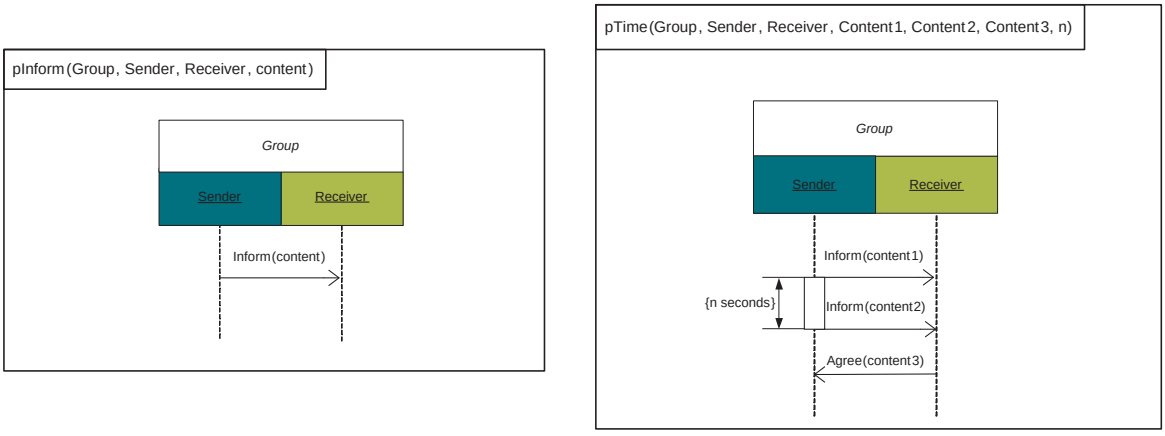


Figure 4.12: SGA Dialogic Specification

Figure 4.12 presents the two generic interaction protocols used for the SGA example: *pInform* and *pTime*.

The protocol *pInform* requires four parameters to be instantiated, respectively a *group name*, a *sender role*, a *receiver role* and a *message content*. This protocol allows to send an *Inform* message from a sender role to a receiver role. This protocol is used in three different steps of the functional specification of the SGA (see Fig. 4.10).

The pTime protocol requires seven parameters, a *group name*, a *sender role*, a *receiver role*, three different *message contents* and a time constraint. As specified in 4.5.4.4 the n value represents the number of seconds between the first *Inform* message containing the parameter *content1* sent from the sender to the receiver and the second one containing the parameter *content2*. Finally, the receiver sends back an *Agree* message to the sender containing the parameter *content3*.

4.5.5 Normative Specification

In the previous sections we have described how to define a set of constraints on the agents through three specifications. During the execution of the organization, the agents will have to play roles within groups as specified in the structural specification (SS), satisfy a set of goals (functional, organizational, interactional or supervisorial) grouped in missions defined in the functional specification (FS) and finally to communicate by the instantiation of generic interaction protocols depicted in the dialogic specification (DiS). We now have to link these specifications in order to express a set of rules concerning one role or one group executing one functional, organizational, supervisorial or interactional (using one or several protocols of the DiS) goal. We therefore define a Normative Specification (NS), as Moise-Inst introduced a normative specification to link its structural, functional and contextual specifications.

The Normative specification is composed of a set of norms specifying relations between the SS, FS and DiS through a bearer, a mission, parameters related to the mission and a deontic operator (permission, obligation or interdiction).

We consider the set $\mathcal{SE}_{SS}$, $\mathcal{M}_{fs}$, $\mathcal{P}_m$ and $\mathcal{S}_{fs}$, grouping respectively the structural entities, the missions, the parameters of the missions and the schemes of the organization. $\mathcal{NS}$ is the set of norms of the organization.
A norm $n \in \mathcal{NS}$ is defined as the following expression: $n = \varphi \rightarrow op(bearer, m, p, s)$ where

$op \in \{obl, per, for\}$
$bearer \in \mathcal{SE}_{SS}$
$m \in \mathcal{M}_{fs}$
$p \in \mathcal{P}_m$
$s \in \mathcal{S}_{fs}$

with:
- φ is an expression of the validity conditions of the norm;
- op is the deontic operator defining an obligation, a permission or an interdiction;
- $bearer$ is a structural entity (role or group) specified in the SS on which the norm is applied;
- m is a functional, organizational, supervisorial or interactional mission specified in the FS specifying

on which action the norm is applied;
- p is the optional set of parameters necessary to the instantiation of the mission m;
- s is the functional scheme of the FS to which belongs the mission m.

φ is the condition which defines the state of the OE in which the norm may be valid. The norm remains valid as long as φ is satisfied. A norm condition could be one of the following expressions:
- *true* which means that the norm is always valid;
- an algorithm-dependent predicate, which can be for instance the algorithm termination condition, based either on a number of iterations expressed as *term==iterations* or on a time constraint expressed as *term==time*.
- a predicate depending on the execution mode which can be batch or graphical and respectively represented as *mode==batch* or *mode==graphical*.

A norm can be fulfilled only once it is valid.

Contrary to the other specifications, it is difficult to provide a graphical representation of the normative specification. Therefore, we represent the norms as shown in Table 4.3.

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	—	obl	role1	m1	parameter1 ∧ parameter2	scheme1
N02	—	obl	role3	m2	parameter2	scheme1
N03	—	per	role2, role3	m4	parameter3	scheme2
N04	—	obl	role4	m5	parameter4	scheme3

Table 4.3: Example of NS Representation

4.5.5.1 Link with the Structural Specification

The *bearer* of a norm is a structural entity on which this norm is applied. A norm can be applied on a role or a group. When the bearer is a group, all the roles belonging to this group have to respect the norm. The agent playing one of these roles on which the norm is applied has to behave in such a way that the norm is not violated.

No matter the operator of the norm, if the bearer of a norm is a role which has inheriting roles, then the latter will also have to respect the norm according to the operator.

The same way, no matter the operator of the norm, if the bearer of a norm is a group which has sub-groups, then the latter will also have to respect the norm according to the operator.

4.5.5.2 Link with the Functional Specification

The functional part of the norms refers to the mission on which the norm is applied. This means that the agent or the group of agents which respect this norm will have to get involved in the mission and to satisfy it.

As previously mentioned in 4.5.3 MAS4EVO extends the FS of Moise+ by allowing the use of functional missions, but also organizational, supervisorial and interactional missions. No matter the type of mission, the expression of a norm is similar to the definition in 4.5.5.

4.5.5.3 Link with the Dialogic Specification:

MAS4EVO normative specification provides normative rules which allow to link one or several roles of the SS to an interactional mission which corresponds to the instantiation of one protocol of the DiS. The only difference is that the multiple bearers can be involved in a norm. The following norm N05 is an example in which both EvoBuilder and EvoMember roles of the DAFO group have the permission to fulfil an interaction mission of the Fabric scheme, in that case mission m5. In this interaction mission, the EvoBuilder role in uses the pInform protocol to send a "Start" message to the EvoMember role.

> N05 = *per(EvoBuilder, EvoMember, m5, FabricScheme)*

4.5.5.4 Normative Specification Example

As for the three other specifications, the following presents the normative specification of the SGA using MAS4EVO.

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	PSA	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	term==iterations	obl	EvoBuilder, EvoMember	m5	—	Fabric
N06	term==time	obl	EvoBuilder, EvoMember	m6	n	Fabric
N07	true	obl	Solver	m7	—	Optimization
N08	true	obl	Solver, Observer	m8	—	Optimization
N09	mode==batch	obl	Observer	m9	—	Observation
N10	mode==graphical	obl	Observer	m10	—	Observation

Table 4.4: Normative Specification of the SGA

The first norm (N01) specifies that the agent playing the Evobuilder role has the obligation to achieve the mission m1 of the Fabric scheme. This means that an agent of the type EvoAgent has to

be created.

The second norm (N02) obliges the PSA to fulfil the mission m2 of the Fabric Scheme which consists in the organizational action adoptRole(EvoMember, DAFO). This means that the PSA has to take the role EvoMember in the group DAFO. The EvoMember is now capable of communicating with the other agents of the DAFO group (Observer and EvoBuilder).

The third norm (N03) obliges the EvoBuilder and the EvoMember to fulfil the mission m3 of the Fabric Scheme which consists in the interactional goal pInform(DAFO, EvoBuilder, EvoMember, Observer, EvoParameters). This means that in the DAFO group, the EvoBuilder has to send the EvoParameters to the EvoMember and to the Observer. The EvoMember and the Observer have now the necessary information concerning the optimization problem and the EvoMember additionally knows the GA it is part of (in that case a SGA) and the parameters to run its generational GA.

The norm N04 is an obligation for the EvoMember to perform the mission m4 of the Fabric scheme which consists in two organizational actions. The first action (createsubGroup(SGA, DAFO)) means that the EvoMember has to create a subgroup of the DAFO group called SGA and the second action (adoptRole(Solver, SGA)) means that the same EvoMember has to take the role Solver in this new SGA group. This group creation and role adoption depend on the GA specified in the parameters previously received (here a SGA).

The norms N05 and N06 are both obligations given to the EvoBuilder and the EvoMember to fulfil an interaction mission of the Fabric scheme. The norm N05 has a validity condition *term==iterations* which means that this norm will be valid if the algorithm's termination condition is based on a number of iterations (i.e. a number of generations or a number of fitness function evaluations). Norm N05 consists in sending a start message from the EvoBuilder to the EvoMember. The norm N06 has a validity condition *term==time* which means that this norm will be valid if the algorithm's termination condition is based on a time constraint of value n as specified in the parameters p. Norm N06 consists for the EvoBuilder in repetitively sending a "start" message, waiting for n seconds, sending a "stop" message and waiting for the "ready" message of the EvoMember.

The norm N07 is an obligation for the Solver to achieve the functional mission m7 of the Optimization scheme, i.e. to run its SGA.

Norm N08 is an obligation for the Solver and the Observer to fulfil the mission m8 of the Optimization Scheme which consists in the interactional goal pInform(DAFO, Solver, Observer, BestIndividual). This means that in the DAFO group, the Solver has to send its BestIndividual to the Observer.

Norms N09 and N10 are obligations for the Observer to achieve the functional missions m9 and m10 of the Observation scheme. The norm N09 has a validity condition *mode==batch* which means that this norm will be valid if the algorithm is run in a batch mode while the norm N10 and its validity condition *mode==graphical* will be valid if the algorithm is run is a graphical mode.

4.6 CGAs Organizational Model

In the preceding section we have described the different specifications provided by MAS4EVO and we illustrated their use to model a simple GA within DAFO. In the coming section, we will demonstrate that through a few changes in these specifications (structural, functional, dialogic and normative) it is possible to model existing CGAs, in our case CCGA and LCGA respectively introduced in 2.8 and 2.2.5.

4.6.1 CCGA Model

This section provides the model of the CCGA in DAFO using MAS4EVO. CCGA is a cooperative CGA using a complete graph as topology of communication between the different subpopulations. It will therefore be necessary to have several EvoAgents (at least 3) to represent these subpopulations. Since all PSAs will communicate with each other, they will all play roles in the same group (i.e. CCGA group) as described in section 4.13.

Two variants of CCGA exist, CCGA-1 in which only the best individual is exchanged between the subpopulations and CCGA-2 in which the best and one random individual are exchanged between the subpopulations. This will be tackled in the functional specification as new possible parameters in the pInform interaction protocol as described in 4.6.1.2.

A specific feature of CCGA is the round-robin process realized in each generation of the algorithm. In this process only one subpopulation after the other is active at one time receiving the individual(s) from all the other subpopulations to evaluate its own individuals. This will be tackled at the structural level by adding two new roles, Consumer and *Producer* (see 4.13), and at the functional level by introducing a supervisorial goal with which PSAs can monitor if it is their turn to become active (i.e. Consumer) and organizational goals which will allow the PSAs to change role (i.e. switch from Producer to Consumer and the opposite) 4.6.1.2.

4.6.1.1 Structural Specification

Figure 4.13 represents the graphical representation of the structural specification of the CCGA using MAS4EVO. The root group is the *DAFO* group which is composed of a single group *CCGA* (cardinality "1..1"). The *DAFO* group contains one *EvoBuilder* role (cardinality "1..1"), one *Observer* role (cardinality "1..1") and all the *EvoMember* roles (cardinality "All"). The *CCGA* group is composed of all the *Solver* roles (cardinality "All") which inherit from the *EvoMember* role. The same way, the

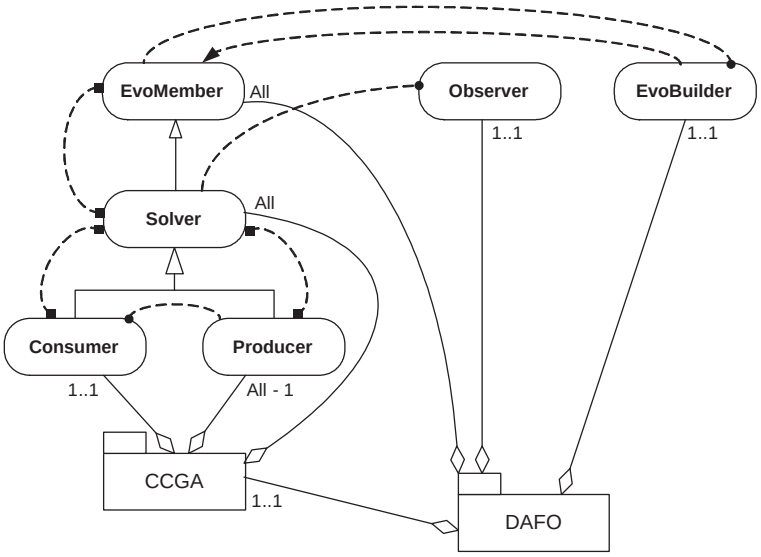


Figure 4.13: CCGA Structural Specification

CCGA group is composed of one *Consumer* role (cardinality "1..1") and *Producer* roles (cardinality "All - 1"), which both inherit from the *Solver* role. This means that all the agents playing the *Solver* role except one will play the *Producer* role, the last agent playing the *Consumer* role (i.e. being the "active" subpopulation).

The *EvoBuilder* role possesses an authority link on the *EvoMember* role since it will control the lifecycle of the *EvoMember* (e.g. instantiation). The *EvoMember* possesses a communication link with the *EvoBuilder* role which will be used to provide information concerning its lifecycle (e.g. ready to compute). The *Solver* role also has a communication link with the *Observer* role so as to send results of its computation process (e.g. the best individual per generation). The *EvoMember* role and the *Solver* role have an intra-group compatibility link which means that the same agent can play both roles in the same instance of the *DAFO* group (we do not mention the *CCGA* group since it is a subgroup of the *DAFO* group). The *Producer* role has a communication link with the *Consumer* role which permits to transmit individuals (i.e. best or best and random for either CCGA-1 or CCGA-2). Finally the *Consumer* and *Producer* roles have an intra-group compatibility link with the *Solver* role which means that the same agent can play both *Consumer* and *Solver* roles or both *Producer* and *Solver* roles in the same instance of the *CCGA* group.

4.6.1.2 Functional Specification

Figure 4.14 represents the functional specification of the CCGA using MAS4EVO. As for the SGA, the specification is built on four social schemes with the same root goals. A functional scheme (considered as the main scheme of the FS) which root goal is to run the DAFO framework (*gDafo*), a fabric scheme which root goal is to initialize the problem solving agent (*gOrganize*), an optimization scheme

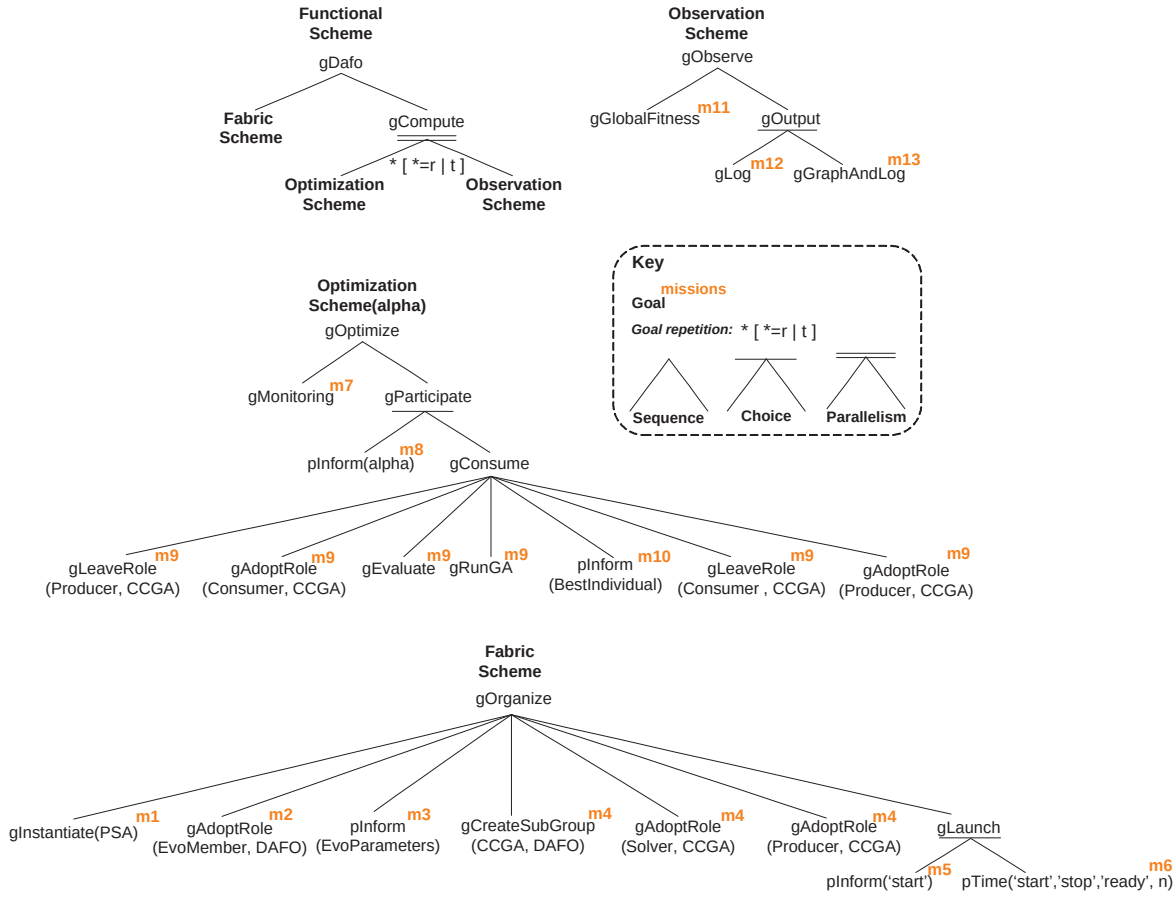


Figure 4.14: Functional Specification of the CCGA

which root goal is to optimize a problem (*gOptimize*) and an observation scheme which root goal is to provide an output of the results obtained by the GA (*gOutput*). All these root goals are artificial goals and consequently are not attached to a mission.

The functional scheme is similar to the SGA's one (see 4.10).

The *Fabric Scheme* is also similar to the SGA's one except that a CCGA group is created instead of a SGA group and the *Producer* role is adopted in the CCGA group addition to the *Solver* one.

The *Optimization Scheme* has more modifications due to the exchange of information between subpopulations and mostly to the round-robin process in which one PSA after the other takes the *Consumer* role instead of its *Producer* role. The artificial root goal *gOptimize* is therefore satisfied once the supervisorial goal *gMonitoring* and the artificial goal *gParticipate* are sequentially satisfied. *gMonitoring* checks if the agent has to keep its *Producer role* or if it has to take a *Consumer role*. This will respectively involve to fulfil the interactional goal *pInform(alpha)* in which a message containing

alpha is sent, which is can be the best individual or the best and a random individual, or to fulfill the artificial goal *gConsume*. This goal is satisfied once the following goals are sequentially satisfied:
- *gLeaveRole(CCGA, Producer)* organizational goal in which the *Producer* role played in the *CCGA* group is quit,
- *gAdoptRole(CCGA, Consumer)* organizational goal in which the *Consumer* role is taken in the *CCGA* group,
- *gEvaluate* the functional goal in which all individuals of a population are evaluated,
- *gRunGA* functional goal running one generation of a SGA,
- *pInform(BestIndividual* interactional goal in which a message containing the best individual is sent,
- *gLeaveRole(CCGA, Consumer)* organizational goal in which the *Consumer* role played in the *CCGA* group is quit,
- *gAdoptRole(CCGA, Producer)* organizational goal in which the *Producer* role is taken in the *CCGA* group.

Finally *Observation Scheme* differs from the SGA one only by the addition of the functional goal *gGlobalFitness*. This goal is executed before the artificial goal *gOutput* and allows to calculate the global fitness based on the received individuals. The remaining goals are thus similar to the SGA's (see 4.10).

Id.	Goals of the mission	Card.	Description
m1	gInstantiate(PSA)	1..1	Instantiate the Problem Solving Agents
m2	gAdoptRole(EvoMember)	3..n	Adopts the EvoMember role in the EvoFramewok group
m3	pInform(EvoParameters)	5..n	Sends the parameters
m4	gCreateSubGroup(CCGA), gAdoptRole(CCGA,Solver), gAdoptRole(CCGA,Producer)	3..n	Creates CCGA group and adopts the Solver and Producer roles in it
m5	pInform(start)	4..n	Sends a start message
m6	pTime(start, stop, ready, n)	4..n	Sends start and stop messages with a time interval n
m7	gMonitoring	3..n	Monitors if keeps Producer role or switch to Consumer
m8	pInform(alpha)	4..n	Sends individuals alpha (best or best and random)
m9	gLeaveRole(CCGA,Producer), gAdoptRole(CCGA,Consumer), gEvaluate, gRunGA, gLeaveRole(CCGA,Consumer), gAdoptRole(CCGA,Producer)	3..n	Takes the Producer role instead of the Consumer, evaluates the indiv. of a population, runs a generation of a SGA and takes back the Consumer role instead of the producer
m10	pInform(BestIndividual)	4..n	Sends the best indiv. of a generation of the SGA
m11	gGlobalFitness	1..n	Calculates the global fitness
m12	gLog	1..1	Logs the best and average fitness
m13	gGraphAndLog	1..1	Logs the best and average fitness and print the corresponding graphs

Table 4.5: Missions definition of the FS of the CCGA

The missions defined for the CCGA are reported in Table 4.5. The Fabric Scheme groups the missions *m1*, *m2*, *m3*, *m4*, *m5* and *m6* concerning the instantiation of the OE and the startup of the computation process.

m1 has a cardinality (1..1) since respectively only one fabric agent can be involved in this mission. *m2* and *m4* have a cardinality (3..n) since at least three problem solving agents (PSAs) will achieve these missions. *m3*, *m5* and *m6* have a cardinality (4..n) since they include an interactional goal which will involve the fabric agent and at least three problem solving agents.

The same way, the optimization scheme groups the missions of the related to the CCGA computation. The supervisorial goal *gMonitoring* is set in a mission (*m7*), the interactional goal *pInform(alpha)* is set in a mission *m8*, the organizational goals *gLeaveRole(CCGA, Producer)*,*gAdoptRole(CCGA, Consumer)*, *gLeaveRole(CCGA, Consumer)*, *gAdoptRole(CCGA, Producer)*, the functional goal *gEvaluate* in which all individuals of a population are evaluated and the functional goal *gRunGA* running one generation of a SGA are all grouped in the same mission *m9*. The interactional goal *pInform(BestIndividual)* is set in a mission (*m10*).

Finally, the Observation scheme groups the missions related to the observation of the computation, *m11* which contains the functional goal *gGlobalFitness* and *m12, m13*, which either log the best fitness received and the calculated average fitness or log the best fitness received, the calculated average fitness and print the corresponding graphs.

4.6.1.3 Dialogic Specification

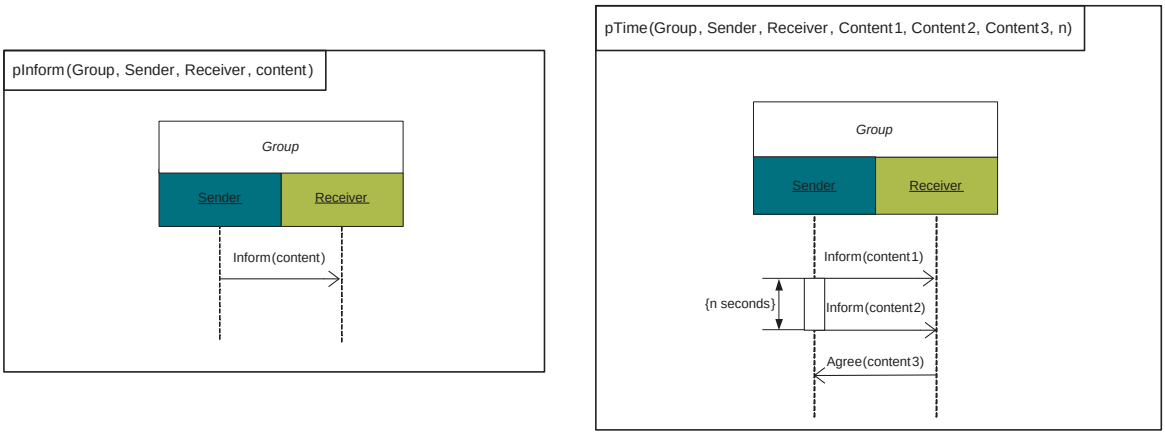


Figure 4.15: CCGA Dialogic Specification

As can be seen, the dialogic specification of the CCGA is similar to SGA's (see 4.12). Indeed, only the *pInform* and the *pTime* generic interaction protocols are required as shown in the FS (see Fig. 4.14).

4.6.1.4 Normative Specification

The following table presents the normative description of the CCGA using MAS4EVO. We will provide a textual description for the norms which differ from the ones defined for the SGA (see 4.5.5.4), which are N04 , N07 , N08 and N11. N1, N2, N3, N5 and N6 are similar to the same norms of SGA. N09 for the CCGA is similar to N07 for the SGA, and the same applies for norms N10 similar to N08, N12 similar to N09 and finally N13 similar to N10.

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	EvoAgent	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	term==iterations	obl	EvoBuilder, EvoMember	m5	—	Fabric
N06	term==time	obl	EvoBuilder, EvoMember	m6	n	Fabric
N07	true	obl	Solver	m7	is true ? (switch to Consumer)	Optimization
N08	true	obl	Producer, Consumer	m8	alpha	Optimization
N09	true	obl	Solver	m9	—	Fabric
N10	true	obl	Producer, Observer	m10	—	Optimization
N11	true	obl	Observer	m11	—	Observation
N12	mode==batch	obl	Observer	m12	—	Observation
N13	mode==graphical	obl	Observer	m13	—	Observation

Table 4.6: Normative Specification of the CCGA

The norm N04 is an obligation for the EvoMember to perform the mission m4 of the Fabric scheme which consists in two organizational actions. The first action (createsubGroup(SGA, DAFO)) means that the EvoMember has to create a subgroup of the DAFO group called CCGA and the second action (adoptRole(Solver, CCGA)) means that the same EvoMember has to take the role Solver in this new CCGA group.

The norm N07 is an obligation for the Solver to monitor if it is its turn to play the Producer role or not.

Norm N08 is an obligation for the Producer and the Consumer to fulfil the mission m8 of the Optimization Scheme which consists in the interactional goal pInform(CCGA, Producer, Consumer, alpha). This means that in the CCGA group, the Producer has to send its alpha individuals (best or best and random) to the Consumer.

The norm N09 is an obligation for the Solver to achieve the mission m9 of the Optimization scheme.

Norm N11 is an obligation for the Observer to fulfil the functional mission m11.

4.6.2 LCGA Model

This section provides a model of the LCGA with a ring topology using MAS4EVO. LCGA is a competitive CGA with no restriction concerning the topology of communication between the different subpopulations. Due to the optimization problems we tackled in chapter 7 and chapter 8, we modeled a LCGAs using ring or complete graph topologies. We therefore chose to use topologies based on a ring, in which it is possible to augment the number of neighbors, from 1 to n-1 (n being the number of PSAs) as illustrated in the following figure.

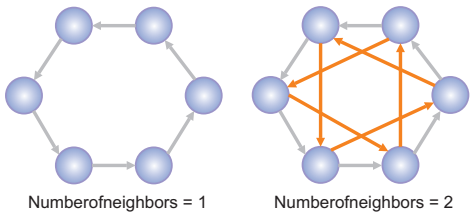


Figure 4.16: LCGA Topologies

As for the CCGA, it will be necessary to have several EvoAgents (at least 3) to represent the subpopulations. Since in these possible topologies one PSA will communicate with one or several neighbors, a new type of group is introduced, *Solving Unit*, which will contain one *Producer* role and one to several *Consumer* roles. In order to create a ring, the same PSA will play the *Consumer* role in one *Solving Unit* group and the *Producer* role in the next *Solving Unit* group. These new group and cardinalities are described in 4.17.

Contrary to CCGA, in LCGA there is no synchronous exchange of individuals between the subpopulations and thus no monitoring of the organization and no modification of the OE (i.e. no switching from Producer to Consumer roles and vice versa). The individuals exchanged between subpopulations are also different since in LCGA random individuals are sent contrary to the best or best and random of the CCGA. This will be taken into consideration with a new parameter in the pInform interaction protocol (i.e. random).

Another difference is the collaborative process introduced in LCGA. Once one subpopulation has calculated the fitness values of its individuals, based on the individuals of its neighbor, these fitness values are sent to the same neighbor population which will use these fitness values to reevaluate its own individuals (typically the average between the fitness it calculated and the fitness it received). This will be tackled in the functional specification as a new functional goal (gReEvaluate) and new possible parameters in the pInform interaction protocol (i.e. fitnesses) as described in 4.6.2.2.

4.6.2.1 Structural Specification

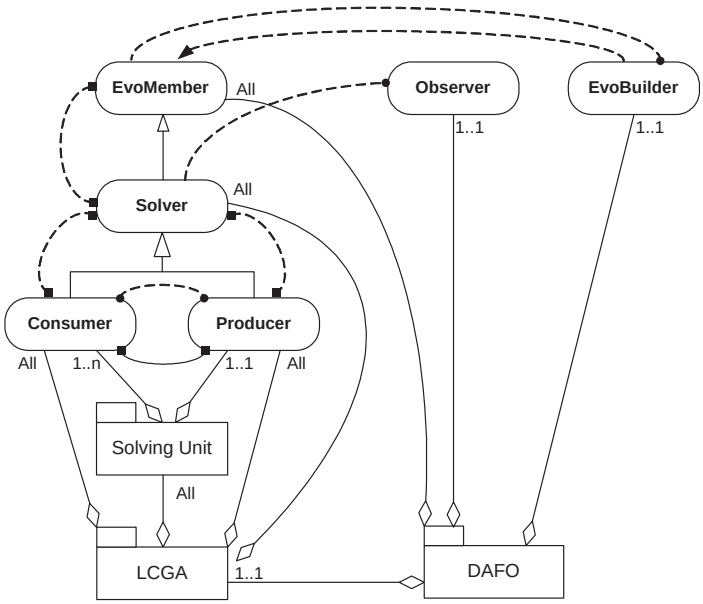


Figure 4.17: LCGA Structural Specification

Figure 4.17 represents the graphical representation of the structural specification of the CCGA using MAS4EVO. The root group is the *DAFO* group which is composed of a single group *LCGA* (cardinality "1..1"). The *DAFO* contains one *EvoBuilder* role (cardinality "1..1"), one *Observer* role (cardinality "1..1") and all the *EvoMember* roles (cardinality "All"). The *LCGA* group is composed of all the *Solver* roles (cardinality "All") which inherits from the *EvoMember* role. The difference with the CCGA in terms of group lies in the addition of the *Solving Unit* groups which are all contained in the *LCGA* group. Each *Solving Unit* group contains one *Producer* role (cardinality "1..1") and one to several *Consumer* roles (cardinality "1..n") which inherits from the *Solver* role. The *Consumer* and *Producer* roles are also all contained in the *LCGA* group (cardinality "All").

The *EvoBuilder* role possesses an authority link on the *EvoMember* role since it will control the lifecycle of the *EvoMember* (e.g. instantiation). The *EvoMember* possesses a communication link with the *EvoBuilder* role which will be used to provide information concerning its lifecycle (e.g. ready to compute). The *Solver* role also has a communication link but with the *Observer* role so as to send results of its computation process (e.g. the best individual per generation).

The first difference with the CCGA concerns the intra-group communication link between the *Consumer* and the *Producer* roles which is in both directions for the LCGA (due to the feedback of the *Consumer* concerning the fitness values obtained) while in CCGA this link was only from the

Producer to the *Consumer*.

Another difference is the inter-group compatibility link between the *Producer* and the *Consumer* roles which means that one agent can not play both roles in the same instance of the *Solving Unit* group but it can in different instances.

4.6.2.2 Functional Specification

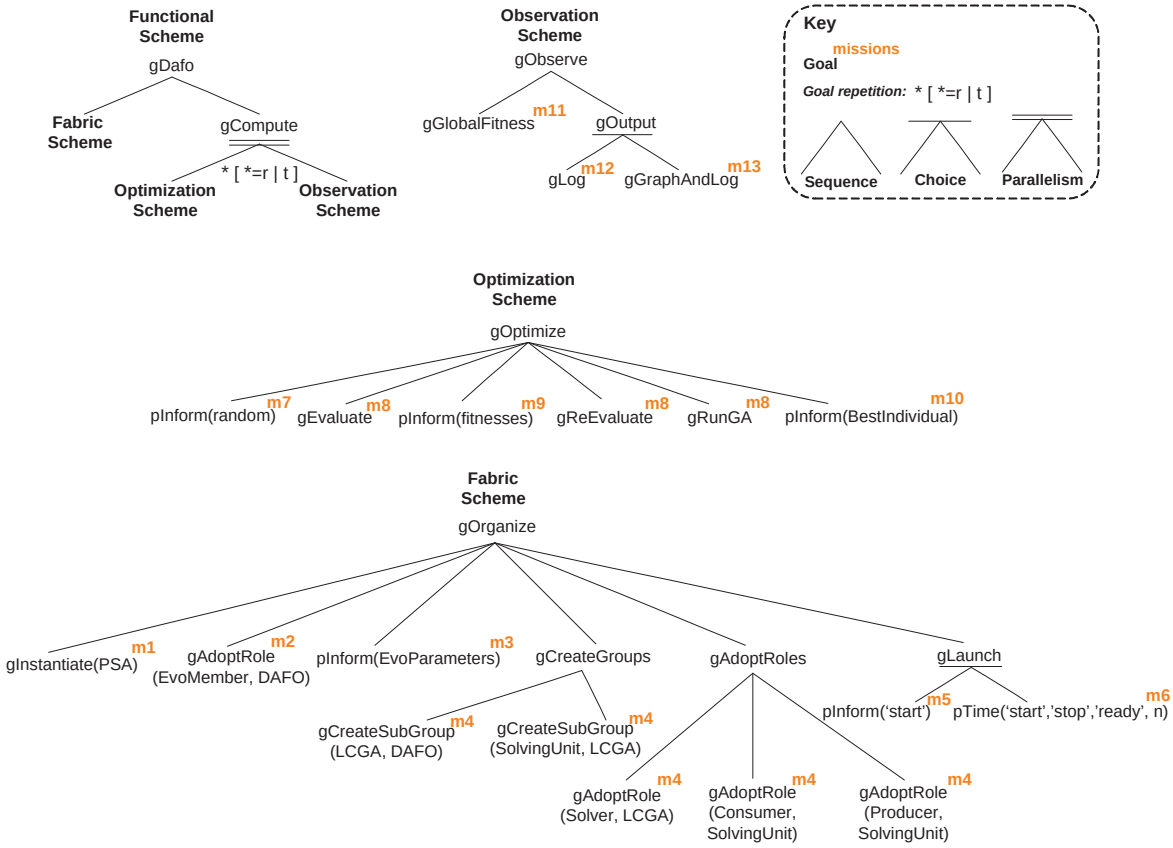


Figure 4.18: Functional Specification of the LCGA

Figure 4.18 represents the functional specification of the LCGA using MAS4EVO. As for the SGA and the CCGA, the specification is built on four social schemes, a functional scheme (considered as the main scheme of the FS) which root goal is to run the DAFO framework (*gDafo*), a fabric scheme which root goal is to initialize the problem solving agent (*gOrganize*), an optimization scheme which root goal is to optimize a problem (*gOptimize*) and an observation scheme which root goal is to provide an output of the results obtained by the GA (*gOutput*). All these root goals are artificial goals and consequently are not attached to a mission.

The functional scheme is similar to the SGA's and the CCGA's ones (see 4.10).

The *Fabric Scheme* of the LCGA is based on the CCGA's one in which a LCGA group is created instead of a CCGA group. It also adds the creation of the new *SolvingUnit* groups (*gCreateSubGroup(SolvingUnit, LCGA)*) and the adoption of the *Producer* and *Consumer* roles in these new groups (*gAdoptRole(Consumer, SolvingUnit)* and *gAdoptRole(Producer, SolvingUnit)*).

The *Optimization Scheme* is more different from the CCGA's since in LCGA there is no need to monitor the organization and no modification of the OE (i.e. no switching from Producer to Consumer roles and vice versa). The artificial root goal *gOptimize* is thus satisfied once the following goals are sequentially achieved:
- *pInform(random)* interactional goal in which a message containing random individuals is sent,
- *gEvaluate* functional goal in which all individuals of a population are evaluated on the optimization problem,
- *pInform(fitnesses)* interactional goal in which a message containing the fitness values obtained with the received individuals are sent. This is due to the collaborative process introduced in LCGA in which the fitness values obtained by one subpopulation on the individuals it received are sent to the neighbor,
- *gReEvaluate* functional goal calculating the new fitness value according to the fitness values received. This allows to reevaluate the individuals sent based on the fitness values received,
- *gRunGA* functional goal running one generation of a SGA,
- *pInform(BestIndividual)* interactional goal in which a message containing the best individual is sent.

Finally *Observation Scheme* is similar to the CCGA's (see 4.6.1.2).

The missions defined for the LCGA are reported in Table 4.7. The Fabric Scheme groups the missions *m1*, *m2*, *m3*, *m4*, *m5* and *m6* concerning the instantiation of the OE and the startup of the computation process.

m1 has a cardinality (1..1) since only one fabric agent can be involved in this mission. *m2* and *m4* have a cardinality (3..n) since at least three problem solving agents (PSAs) will achieve these missions. *m3*, *m5* and *m6* have a cardinality (4..n) since they include an interactional goal which will involve the fabric agent and at least three problem solving agents.

The same way, the optimization scheme groups the missions related to the LCGA computation. The interactional goal *pInform(random)* is set in a mission (*m7*), the functional goals *gEvaluate*, *gReEvaluate* and *gRunGA* are set in a mission (*m8*), and the interaction goals *pInform(fitnesses)* and *pInform(bestIndividual)* are set in missions *m9* and *m10*.

Id.	Goals of the mission	Card.	Description
m1	gInstantiate(PSA)	1..1	Instantiate the Problem Solving Agents
m2	gAdoptRole(EvoMember)	3..n	Adopts the EvoMember role in the EvoFramewok group
m3	pInform(EvoParameters)	4..n	Sends the parameters
m4	gCreateSubGroup(LCGA, DAFO), gCreateSubGroup(SolvingUnit, LCGA), gAdoptRole(Solver, LCGA), gAdoptRole(Consumer,SolvingUnit), gAdoptRole(Producer,SolvingUnit)	3..n	Creates the LCGA and the Solving Unit groups and adopts the Solver, Consumer and Producer roles
m5	pInform(start)	4..n	Sends a start message
m6	pTime(start, stop, ready, n)	4..n	Sends start and stop messages to the with a time interval n
m7	pInform(random)	4..n	Sends random individuals
m8	gEvaluate, gReEvaluate, gRunGA	3..n	Evaluates the indiv. of a population, re-evaluates according to the fitness received and runs a generation of a SGA
m9	pInform(fitnesses)	3..n	Sends fitness values
m10	pInform(bestIndividual)	2..2	Sends the best indiv. of a generation of a SGA
m11	gGlobalFitness	1..1	Calculates the global fitness
m12	gLog	1..1	Logs the best and average fitness
m13	gGraphAndLog	1..1	Logs the best and average fitness and print the corresponding graphs

Table 4.7: Missions definition of the FS of the LCGA

Finally, the Observation scheme groups the missions related to the observation of the computation. *m11* contains the functional goal *gGlobalFitness* and *m12, m13*, respectively log the best fitness received and the calculated average fitness or log the best fitness received, the calculated average fitness and prints the corresponding graphs.

4.6.2.3 Dialogic Specification

As for the CCGA, the dialogic specification is similar to the SGA's (see 4.12). Indeed, only the *pInform* and the *pTime* generic interaction protocols are required as shown in the FS (see Fig. 4.18).

4.6.2.4 Normative Specification

The following table presents the normative description of the LCGA using MAS4EVO. We will provide a textual description for the norms which differ from the ones defined for the CCGA (see 4.6.1.4), which are N04 , N07 , N08 and N09.

The norm N04 is an obligation for the EvoMember to perform the mission m4 of the Fabric scheme which consists in two organizational actions. The first two organizational actions createsubGroup(LCGA, DAFO) and createsubGroup(SolvingUnit, LCGA) mean that the EvoMember has to create a subgroup of the DAFO group called LCGA and subgroups of the LCGA group called SolvingUnit. The next three organizational actions oblige the EvoMember to take the role Solver in the LCGA group and the roles Consumer and Producer in two consecutive SolvingUnit groups.

Norm N07 is an obligation for the Producer and the Consumer to fulfil the mission m8 of the Optimization Scheme which consists in the interactional goal pInform(SolvingUnit, Producer, Consumer,

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	EvoAgent	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	term==iterations	obl	EvoBuilder, EvoMember	m5	—	Fabric
N06	term==time	obl	EvoBuilder, EvoMember	m6	n	Fabric
N07	true	obl	Producer, Consumer	m7	—	Optimization
N08	true	obl	Consumer	m8	—	Optimization
N09	true	obl	Consumer, Producer	m9	—	Optimization
N10	true	obl	Producer, Observer	m10	—	Optimization
N11	true	obl	Observer	m11	—	Observation
N12	mode==batch	obl	Observer	m12	—	Observation
N13	mode==graphical	obl	Observer	m13	—	Observation

Table 4.8: Normative Specification of the LCGA

random). This means that in the SolvingUnit group, the Producer has to send its random individuals to the Consumer.

The norm N08 is an obligation for the Consumer to achieve the mission m9 of the Optimization scheme.

Norm N09 is an obligation for the Producer and the Consumer to fulfil the mission m9 of the Optimization Scheme which consists in the interactional goal pInform(SolvingUnit, Consumer, Producer, fitnesses). This means that in the SolvingUnit group, the Consumer has to send its fitness values to the Producer.

4.7 Conclusion

This chapter introduced the MAS4EVO of the DAFO framework. This description has been done using the vowel approach (AEIO), from the BDI-like agent architecture in 6.2.1, to the environment and the interactions described in 4.3 and finally the organization in 4.5.

In order to meet our requirements, we proposed a new organizational model dedicated to evolutionary optimization and based on Moise+, MAS4EVO. Its structural, functional specifications extending Moise+ were described respectively in 4.5.2 and in 4.5.3 and two new specifications were introduced, a dialogic specification (see 4.5.4) and a normative specification (see 4.5.5). The use of these four specifications was illustrated by the modeling example of a SGA.

Finally, MAS4EVO was used to model two existing CGAs which are available in DAFO, the CCGA and the LCGA. Through these two examples it was demonstrated that with few changes in these specifications it is possible to switch from one CGA to the other.

Chapter 5

Hybrid and Dynamic LCGA Models

Contents

5.1 Introduction

The previous chapter introduced a new organizational model dedicated to evolutionary optimization and its use for modeling two existing coevolutionary genetic algorithms, a cooperative (CCGA) and a competitive (LCGA).

This chapter introduces two new CGAs based on LCGA which have been developed in DAFO, a hybrid one and a dynamic one, respectively called hLCGA (see section 5.2) and dLCGA (see section 5.3). For each of those two algorithms, a general description of their functioning is provided and then their complete organizational model using the four specifications of MAS4EVO (structural, functional, dialogic and normative) is detailed.

5.2 hLCGA : A new hybrid LCGA

As previously mentioned in chapter 2.2.4.1, hybridization of GAs has been a very active research area. However, hybrid coevolutionary genetic algorithms have been rarely tackled, except in [25], in which Son and Bladwick hybridized a CCGA with a hill-climbing algorithm. Due to the few existing researches in this area, we decided to investigate the hybridization of the LCGA with different local search algorithms.

Section 5.2.1 presents a detailed description of hLCGA, the new hybrid LCGA. In section 5.2.2 we describe hLCGA's organizational model using MAS4EVO's organizational specifications.

5.2.1 hLCGA Description

Although it is possible to make a genetic algorithm hybrid in different ways, latest published articles [25] have shown that combining genetic algorithms with local search algorithms are one of the best approaches for improving the results.

For this reason we chose to hybridize LCGA with various local search algorithms that are described below:

- Steepest Ascent Hill Climbing (SAHC): systematically flips each single bit of the chromosome and records all the obtained fitness values. The resulting chromosome with the highest fitness is kept if its fitness value is better than the fitness value of the initial chromosome. If no modified chromosome gives a better result, the algorithm stops.
- Next Ascent Hill Climbing (NAHC): systematically flips each bit from left to right until the resulting fitness increases. If the fitness increased then the flipped bit is kept otherwise it is flipped back. If there is no improvement once at the end of the chromosome, the algorithms stops.

- Random Bit Climbing (RBC) [153][154]: similar to NAHC, except that there is no left to right iteration. Instead a random permutation is generated to determine the order in which bits flips are tested. After flipping every bit in the initial solution string, a new random sequence is chosen for testing the bits and the bit climber again checks every bit for an improvement. If the bit climber has tested every bit and no improvement is found, a local optimum has been reached.
- Dynamic Hill Climbing (DHC): introduced by Yuret in [155], it is based on the following main key heuristics: adjusting the size of probing steps to suit the local nature of the terrain, shrinking when probes do poorly and growing when probes do well and keeping track of the directions of recent successes, so as to probe preferentially in the direction of most rapid ascent.
- Tabu Search (TS) [156]: The tabu search algorithm is built around two important aspects, the tabu list that memorizes the last forbidden moves and the aspiration criteria that allows a tabu status of a tabu move to be overwritten. In case tabu is combined with GA, every individual in the population maintains a tabu list. The tabu list is kept from one generation to the other until the individual is replaced by offspring.

Below is the hybrid LCGA algorithm showing where the local search has been incorporated:

Algorithme 6 : hLCGA

```
gen = 0
foreach player_s do
    Pop_s(gen) = randomly initialized population
    evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
end
while termination condition = false do
    gen = gen + 1
    foreach player_s do
        select Pop_s(gen) from Pop_s(gen - 1) based on fitness
        apply genetic operators to Pop_s(gen)
        evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
        apply local search on a percentage of Pop_s(gen)
    end
end
```

Coming with the local search addition, some new parameters can be fine-tuned: the exchange rate fixing the proportion of the population that will be optimized by the local search (it is also possible to apply local search only on the current best individual) and the choice between a restricted or a complete search. If restricted, the local search algorithm will stop after the first improvement and if complete the local search algorithm will be fully executed.

5.2.2 hLCGA Model

This section provides a model of the hLCGA with a ring topology using MAS4EVO.

The model of this hybrid LCGA is therefore based on the LCGA's presented in 4.6.2. To make the LCGA hybrid, each PSA running a SGA and representing one subpopulations of the LCGA will communicate with another new PSA running one local search algorithm (i.e. the gRunLS functional goal as described in 4.4.2). This topology is represented in Figure 5.1.

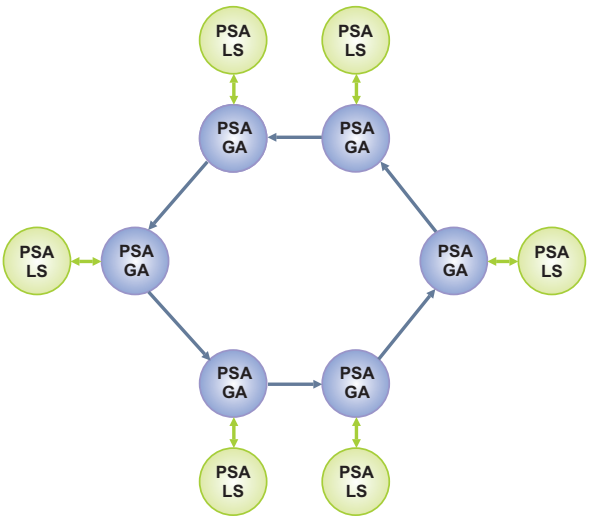


Figure 5.1: hLCGA with ring topology

As for the LCGA, it will be necessary to have several PSA (at least 3) to represent the subpopulations and consequently three additional PSA to run the local search algorithm. In order to allow a communication between these PSAs, a new type of group was added in the SS , *LocalSearchUnit*, in which the PSA running a SGA will play the *Solver* role and the PSA running the LS algorithm will play a new *LocalSearcher* role (see 5.2).

These new *LocalSearchUnit* groups will have to be created and these new *LocalSearcher* roles will have to be adopted by the additional PSAs. This will be tackled in the FS with the addition of the organizational goals gCreateSubGroup(LocalSerach, DAFO) and the gAdoptRole(LocalSearcher, SolvingUnit).

Once the OE instantiated, it is necessary to run the LS algorithm. This is also achieved in the FS by adding the exchange of individual(s) (best individual or population rate defined by the parameter alpha) from the *Solver* role to the *LocalSearcher* with the interactional goal pInform(alpha). Then the LS algorithm is run with the functional goal gRunLS and finally the optimized individual(s) are sent back using the same interactional goal pInform(alpha).

These new goals will be part of new missions which will be attached to roles in new norms of the NS (see 5.2.2.4).

5.2.2.1 Structural Specification

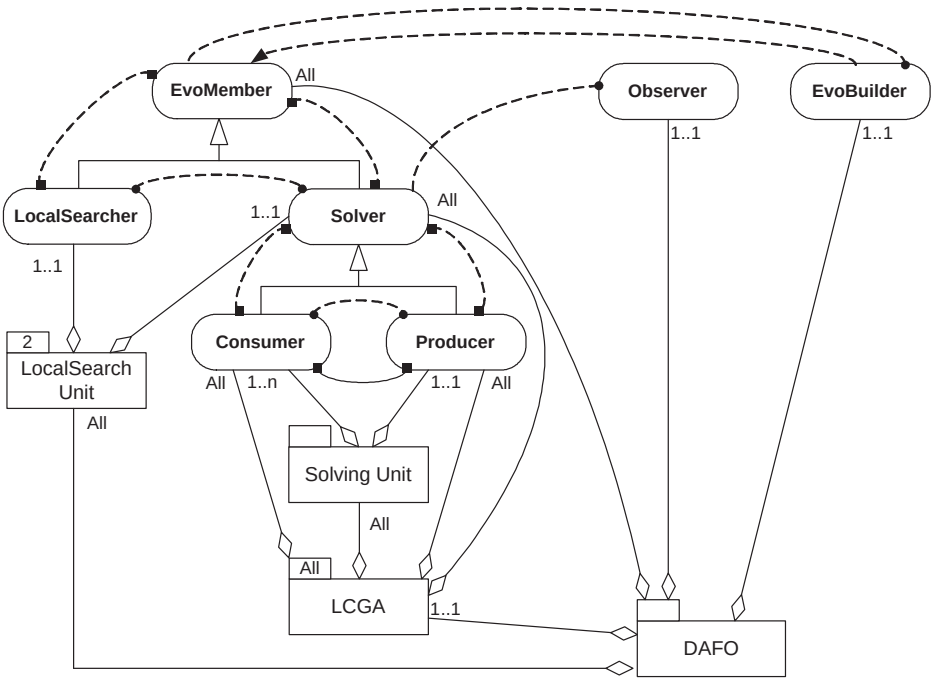


Figure 5.2: hLCGA Structural Specification

The SS of the hLCGA presented in Figure 5.2 is thus similar to the LCGA's SS plus the addition of the new group *LocalSearchUnit* which is a subgroup of the *DAFO* group and of the new role *LocalSearcher*. All the instances of the *LocalSearchUnit* subgroup are contained in the *DAFO* group (cardinality "All").

The *LocalSearch Unit* group contains exactly one *Solver* role and one *LocalSearcher* role (cardinality "1..1") which both inherit from the *EvoMember* role. Those two roles have to be played by two different agents due to the cardinality "2" on the group.

The *LocalSearcher* role and the *EvoMember* role have a new compatibility link which means that the same agent can play both roles in the same instance of the *DAFO* group (we do not mention the *LCGA* group since it is a subgroup of the *DAFO* group).

Finally, a new intra-group communication link between the *LocalSearcher* and the *Solver* roles and vice versa is added. This will allow the exchange of individual(s) necessary for the LS algorithm.

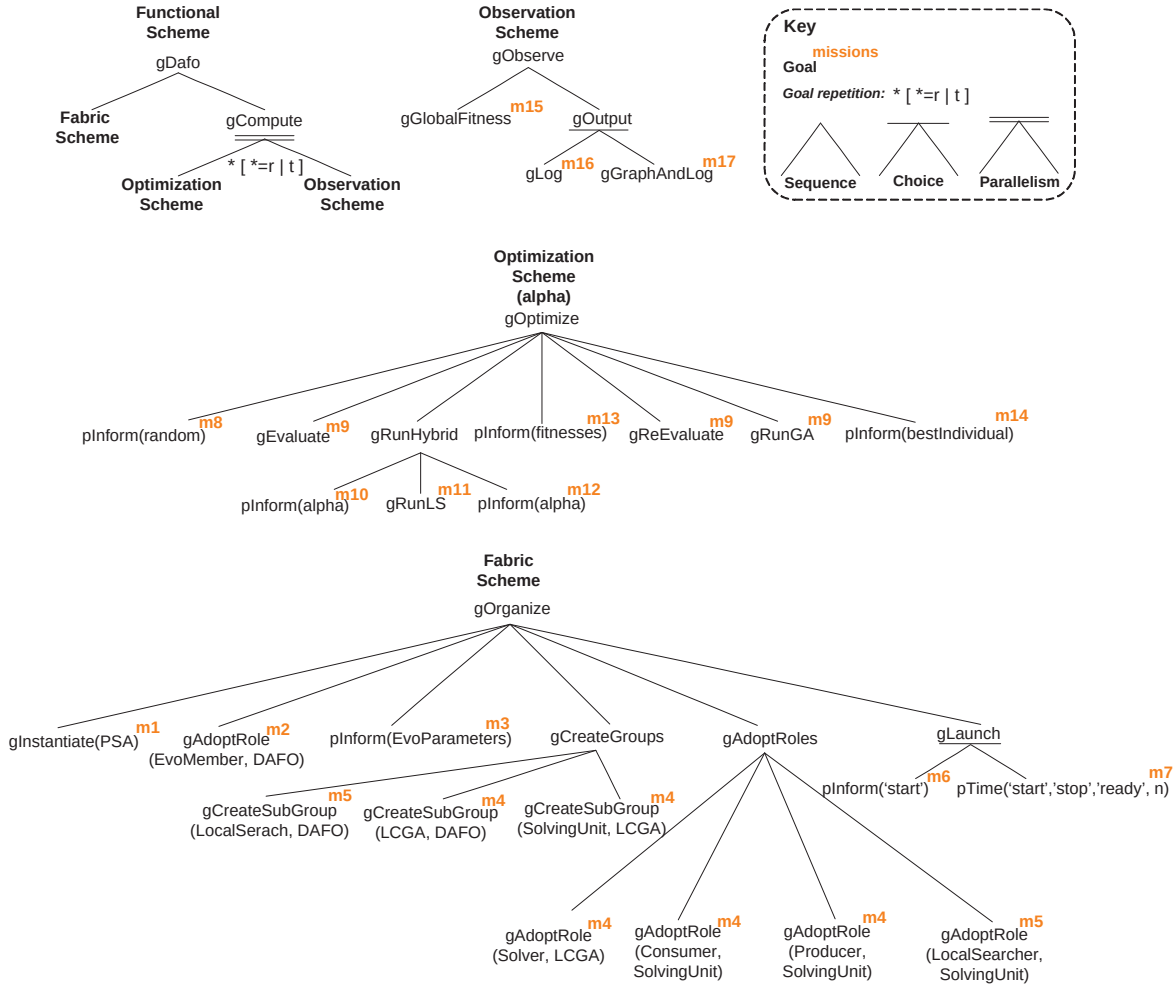


Figure 5.3: Functional Specification of the hLCGA

5.2.2.2 Functional Specification

The FS of the hLCGA represented in Figure 5.3 is also similar to the FS of the LCGA (see 4.6.2.2) plus the addition of the two goals in the *Fabric Scheme* and three goals in the*Optimization Scheme*. The two additional goals of the *Fabric Scheme* are :

- gCreateSubGroup(LocalSerach, DAFO) organizational goal creating the *LocalSearch* group as a sub-group of the *DAFO* group,
- gAdoptRole(LocalSearcher, SolvingUnit) organizational goal in which the *LocalSearcher* role is taken in the *SolvingUnit* group,

The three goals added to the *Optimization Scheme* are:

- gRunHybrid artificial goal used to group the sequence of goals related to the hybridization,
- pInform(alpha) interactional goal sending the *alpha* individual(s) (alpha = best individual or popu-

lation rate),
- gRunLS functional goal running a local search algorithm.

Id.	Goals of the mission	Card.	Description
m1	gInstantiate(PSA)	1..1	Instantiate the Problem Solving Agents
m2	gAdoptRole(EvoMember)	6..n	Adopts the EvoMember role in the EvoFramewok group
m3	pInform(EvoParameters)	7..n	Sends the parameters
m4	gCreateSubGroup(LCGA, DAFO), gCreateSubGroup(SolvingUnit, LCGA), gAdoptRole(Solver, LCGA), gAdoptRole(Consumer,SolvingUnit), gAdoptRole(Producer,SolvingUnit)	3..n	Creates the LCGA and the Solving Unit groups and adopts the Solver, Consumer and Producer roles
m5	gCreateSubGroup(LocalSerach, DAFO), gAdoptRole(LocalSearcher, SolvingUnit)	3..n	Creates the LocalSearch group and adopts the LocalSolver role in it
m6	pInform(start)	4..n	Sends a start message
m7	pTime(start, stop, ready, n)	4..n	Sends start and stop messages with a time interval n
m8	pInform(random)	4..n	Sends random individuals
m9	gEvaluate, gReEvaluate, gRunGA	3..n	Evaluates the indiv. of a population, re-evaluates according to the fitness received and runs a generation of a SGA
m10	pInform(alpha)	3..n	Sends individual(s) (best or population rate)
m11	gRunLS	3..n	Runs one of the available local search alrgorithm
m12	pInform(alpha)	3..n	Sends possibly improved individual(s) (best or population rate)
m13	pInform(fitnesses)	3..n	Sends fitness values
m14	pInform(bestIndividual)	3..n	Sends the best indiv. of a generation of a SGA
m15	gGlobalFitness	1..1	Calculates the global fitness
m16	gLog	1..1	Logs the best and average fitness
m17	gGraphAndLog	1..1	Logs the best and average fitness and print the corresponding graphs

Table 5.1: Missions definition of the FS of the hLCGA

The missions defined for the hLCGA are reported in Table 5.1.
The Fabric Scheme groups the missions *m1*, *m2*, *m3*, *m4*, *m5*, *m6* and *m7* concerning the instantiation of the OE and the startup of the computation process. *m1* has a cardinality (1..1) since only one Fabric Agent can be involved in this mission. *m2* has a cardinality (6..n) since all Problem Solving Agents (genetic and local search) can achieve this mission. *m3* has a cardinality (7..n) since all Problem Solving Agents (genetic and local search) have to receive the EvoParameters. *m4* and *m5* have a cardinality (3..n) since at least three problem solving agents will achieve these missions. *m6* and *m7* have a cardinality (4..n) since they include an interactional goal which will involve the Fabric Agent and at least three Problem Solving Agents.

The same way, the optimization scheme groups the missions related to the LCGA and to the local search algorithm computation. The interactional goal *pInform(random)* is set in a mission (*m8*), the functional goals *gEvaluate*, *gReEvaluate* and *gRunGA* are set in a mission (*m9*), the interaction goals *pInform(alpha)* are set in missions *m10*, *m12*, *pInform(fitnesses)* in mission *m13* and finally *pInform(bestIndividual)* in mission *m14*.

Finally, the Observation scheme groups the missions related to the observation of the computation. *m15* contains the functional goal *gEvaluate* and *m16, m17*, either log the best fitness received and the calculated average fitness or log the best fitness received, the calculated average fitness and prints the corresponding graphs.

5.2.2.3 Dialogic Specification

As for the CCGA and the LCGA, the dialogic specification is similar to the SGA's (see 4.12). Indeed, only the *pInform* and the *pTime* generic interaction protocols are required as shown in the FS (see Fig. 5.3).

5.2.2.4 Normative Specification

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	EvoAgent	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	true	obl	EvoMember	m5	—	Fabric
N06	term==iterations	obl	EvoBuilder, EvoMember	m6	—	Fabric
N07	term==time	obl	EvoBuilder, EvoMember	m7	n	Fabric
N08	true	obl	Producer, Consumer	m8	—	Optimization
N09	true	obl	Consumer	m9	—	Optimization
N10	true	obl	Solver, LocalSearcher	m10	alpha	Optimization
N11	true	obl	LocalSearcher	m11	—	Optimization
N12	true	obl	LocalSearcher, Solver	m12	alpha	Optimization
N13	true	obl	Consumer, Producer	m13	—	Optimization
N14	true	obl	Producer, Observer	m14	—	Optimization
N15	true	obl	Observer	m15	—	Observation
N16	mode==batch	obl	Observer	m16	—	Observation
N17	mode==graphical	obl	Observer	m17	—	Observation

Table 5.2: Normative Specification of the hLCGA

The following table presents the normative description of the hLCGA using MAS4EVO. We will provide a textual description for the norms which differ from the ones defined for the LCGA (see 4.6.2.4), which are N05 , N10 , N11 and N12.

N01, N02, N03, N04 and are similar to the same norms of LCGA. N06 for the hLCGA is similar to N05 for the LCGA, and the same applies for norms N07 similar to N06, N08 similar to N07, N09 similar to N08, and finally N13, N14, N15, N16, N17 in hLCGA are similar to N09, N10, N11, N12, N13 in LCGA.

The norm N05 obliges the EvoMember to fulfil two organizational actions, which are to create a LocalSearchUnit subgroup (createsubGroup(LocalSearchUnit, EvoFramework)) and to take the LocalSearcher in it (roleadoptRole(LocalSearcher, LocalSearchUnit)).

N10 is a obligation for the Solver and the LocalSearcher to fulfil the mission m10 of the Optimization Scheme which consists in the interactional goal pInform(LocalSearchUnit, Solver, LocalSearcher, alpha)), Optimization Scheme). This means that in the LocalSearchUnit group, the Solver has to send its alpha individuals (best or population rate) to the LocalSearcher. The norm N12 obliges the same process but the opposite way (LocalSolver to Solver).

N11 is an obligation for the LocalSolver to fulfil mission m11.

5.3 dLCGA: a new dynamic LCGA

As mentioned in chapter 2.2.4.2, a few previous researches have tackled the adaptive CGAs, also known as dynamic CGAs. We have seen that these related works focused either on the adaptation of the number of populations (see [26]) or on the adaptation of the parameters (see [27] and [28]).

Our contribution consists in building a dynamic LCGA, in which the topology of communication between the population evolves during runtime. Indeed, contrary to CCGA where the topology is fixed (i.e. fully connected graph), using LCGA makes no restriction on the communication graph, since it fully depends on the decomposition of the optimized problem.

Section 5.3.1 presents a detailed description dLCGA, the new dynamic LCGA. In section 5.3.2 we describe dLCGA's organizational model using MAS4EVO's organizational specifications.

5.3.1 dLCGA Description

dLCGA is a new dynamic version of LCGA, which graph of interaction is modified each n generations of the algorithm. The modification is achieved through a cooperative process starting with the first player who randomly chooses a new position in the graph of interaction and informs all the other players of his local decision. The next player in the graph will then randomly choose a new position among the remaining available ones and inform the other players. This process is iteratively executed by all players. Once finished, each player goes to its new position and the algorithm runs again for n generations.

Through this random process, each population exchanges information with different populations during runtime, and thus has to evaluate its individuals using different parts of the solution.

Figure 5.4 shows an example of a dLCGA using a ring topology using a simplified view of an organizational entity (OE). After n generations of the algorithm, all *Evolutionary Agents* leave the roles

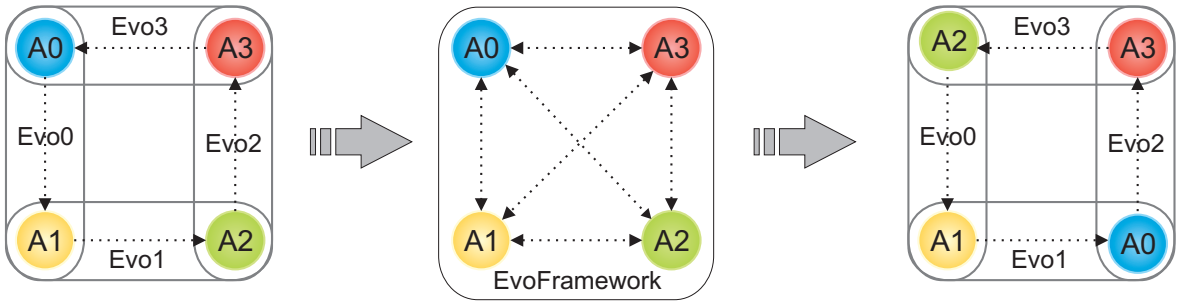


Figure 5.4: dLCGA dynamics

they play in the *Evo* groups. Consequently, they only play a role in the "base" group *EvoFramework* and this way they communicate all together in order to define the groups in which each of them will play a role. Once they all know their new location, they take their roles in the newly defined groups.

5.3.2 dLCGA Model

This section provides a model of the dLCGA with a ring topology using MAS4EVO. As for the hLCGA, the dLCGA's model is based on the LCGA's presented in 4.6.2.

Since the topology changes consist in moving the PSAs on the ring, this only affects the organizational entity (OE) and not the organizational specification (OS). Therefore, the structural specification of the dLCGA is similar to the LCGA's.

The reorganization process is started after a predefined number of generation in each subpopulation. This means that each PSA will have to monitor its computation in order to verify if the condition is met or not, this will be achieved by a new supervisorial goal *gMonitoring*. Once this condition met, each PSA will leave its *Consumer* and *Producer* roles in its *SolvingUnit* groups, which will be done with two new organizational goals *gLeaveRole(Producer, SolvingUnit)* and *gLeaveRole(Producer, SolvingUnit)*. Then the PSAs will have to negotiate with each other to find their new groups. This will be achieved with a new interactional goal pNegotiate, which implies the definition of a new interaction protocol in the dialogic specification (see 5.3.2.3). Finally the PSAs will adopt their *Consumer* and *Producer* roles in their newly defined *SolvingUnit* groups using the *gAdoptRole(Producer, SolvingUnit)* and *gAdoptRole(Producer, SolvingUnit)* organizational goals. All these new goals will be grouped in a new social scheme called *Reorganization Scheme*.

These new goals will be part of new missions which will be attached to roles in new norms of the NS (see 5.3.2.4).

5.3.2.1 Structural Specification

As already mentioned, the SS of the dLCGA is similar to the SS of the LCGA (see 4.17).

5.3.2.2 Functional Specification

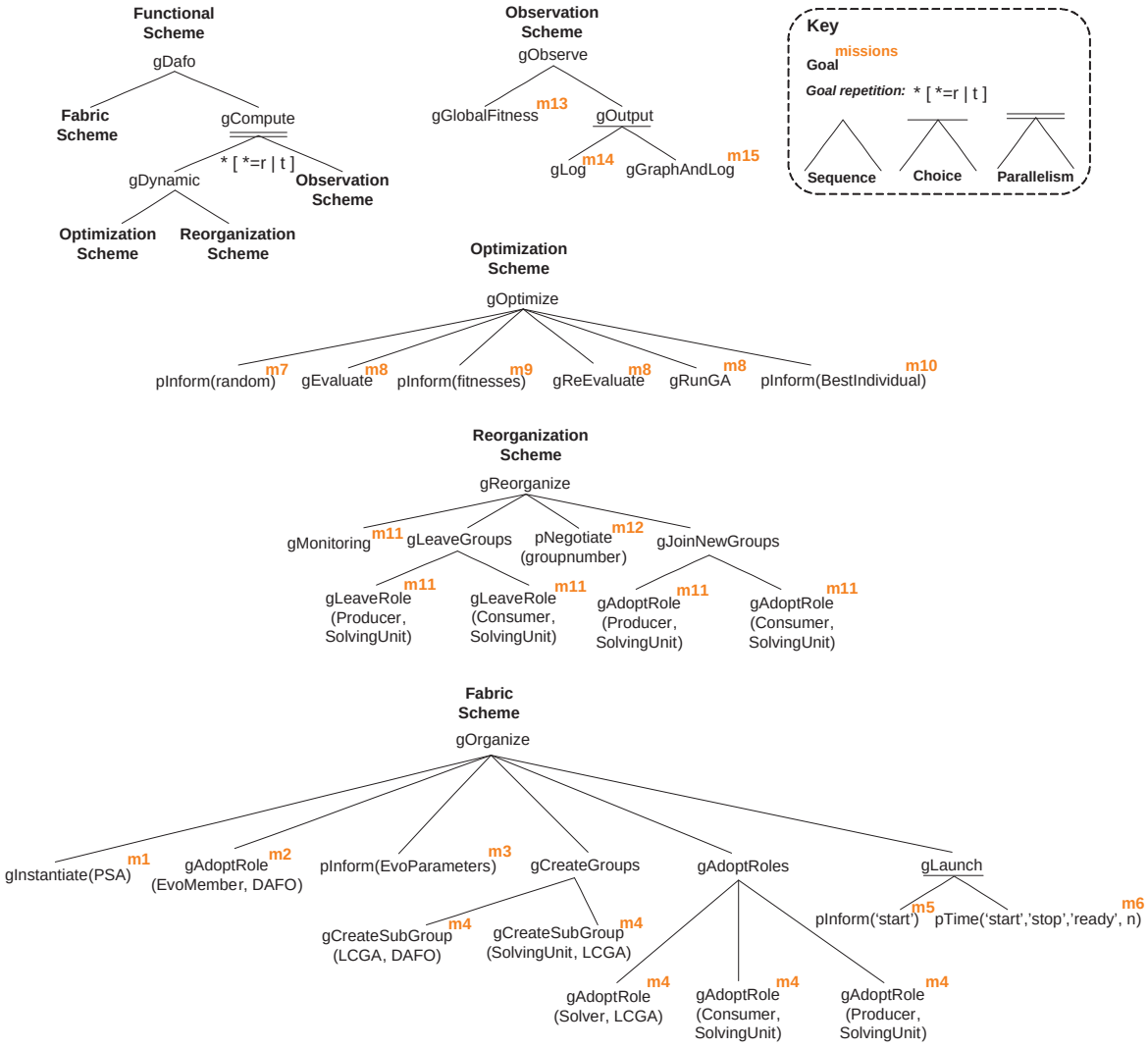


Figure 5.5: Functional Specification of the dLCGA

Figure 5.5 represents the functional specification of the dLCGA within DAFO. As stated before, the specification is no more built on four social schemes but one five, due to the introduction of a new reorganization scheme.

The functional scheme is modified to take into account this additional social scheme. The optimization scheme has been replaced by a new artificial goal *gDynamic*. *gDynamic* is fulfilled when the *Optimization Scheme* and the *Reorganization Scheme* are sequentially realized.

The artificial root goal *gReorganize* of the new *Reorganization Scheme* is satisfied when the supervisorial goal *gMonitoring* observing if the condition activating the scheme is fulfilled, the organizational role *gLeaveGroups* in which the *Producer* and *Consumer* roles of the *Solving Unit* groups are left, the *pNegotiate* interaction protocol in which the negotiation is conducted to find the new *Solving Unit* groups and the *gJoinNewGroups* organizational goal in which the the *Producer* and *Consumer* are adopted in the newly assigned *Solving Unit* groups, are sequentially satisfied.

Id.	Goals of the mission	Card.	Description
m1	gInstantiate(PSA)	1..1	Instantiate the Problem Solving Agents
m2	gAdoptRole(EvoMember)	3..n	Adopts the EvoMember role in the EvoFramewok group
m3	pInform(EvoParameters)	4..n	EvoBuilder ends the parameters to the EvoMember
m4	gCreateSubGroup(LCGA, DAFO), gCreateSubGroup(SolvingUnit, LCGA), gAdoptRole(Solver, LCGA), gAdoptRole(Consumer,SolvingUnit), gAdoptRole(Producer,SolvingUnit)	3..n	Creates the LCGA and the Solving Unit groups and adopts the Solver, Consumer and Producer roles
m5	pInform(start)	4..n	EvoBuilder sends a start message to the EvoMember
m6	pTime(start, stop, ready, n)	4..n	EvoBuilder sends start and stop messages to the EvoMember
m7	pInform(random)	4..n	Producer send individual(s) to Consumer
m8	gEvaluate, gReEvaluate, gRunGA	3..n	Evaluates the indiv. of a population, re-evaluates according to the fitness received and runs a generation of a SGA
m9	pInform(fitnesses)	3..n	Consumer sends fitness values to the Producer
m10	pInform(bestIndividual)	4..n	Consumer sends the best indiv. of a generation to the Observer
m11	gMonitoring, gLeaveRole(Producer, SolvingUnit), gLeaveRole(Producer, SolvingUnit), gAdoptRole(Producer, SolvingUnit), gAdoptRole(Producer, SolvingUnit)	3..n	Monitors and waits until the reorganization criterion is reached, leaves the Solving Unit groups, joins the new Solving Unit Groups
m12	pNegociate	3..n	Negotiate their new Solving Unit groups
m13	gGlobalFitness	1..1	Calculates the global fitness
m14	gLog	1..1	Logs the best and average fitness
m15	gGraphAndLog	1..1	Logs the best and average fitness and print the corresponding graphs

Table 5.3: Missions definition of the FS of the dLCGA

The missions defined for the dLCGA are reported in Table 5.3. The Fabric Scheme groups the missions *m1*, *m2*, *m3*, *m4*, *m5* and *m6* concerning the instantiation of the OE and the startup of the computation process. *m1* has a cardinality (1..1) since respectively only one fabric agent can be involved in this mission. *m2* and *m4* have a cardinality (3..n) since at least three problem solving agents can achieve these missions. *m3*, *m5* and *m6* have a cardinality (4..n) since they include an interactional goal which will involve the the Fabric Agent and at least three Problem Solving Agents.

The same way, the optimization scheme groups the missions of the optimization process of the related to the LCGA. The interactional goal *pInform(random)* is set in a mission (*m7*), the functional goals *gEvaluate*, *gReEvaluate* and *gRunGA* are set in a mission (*m8*), and the interaction goals*pInform(fitnesses)* and *pInform(bestIndividual)* are set in missions *m9* and *m10*.

The reorganization scheme groups the missions of the reorganization process of the Problem Solving Agents. The mission *m11* groups the supervisorial goal *gMonitoring* and the organizational goals *gLeaveGroups* and *gJoinNewGroups*, while the mission *m12* contains the interactional goal *pNegotiate*.

Finally, the Observation scheme groups the missions related to the observation of the computation. *m13* contains the functional goal *gGlobalFitness* and *m14, m15*, either log the best fitness received and the calculated average fitness or log the best fitness received, the calculated average fitness and prints the corresponding graphs.

5.3.2.3 Dialogic Specification

Contrary to the other algorithms specified, a new interaction protocol has been necessary in the reorganization scheme: *pNegotiate*. *pNegotiate* is represented in Figure 5.6. This protocol requires three parameters to be instantiated, respectively a *group name*, a *sender/receiver role* and a *message content*. As mentioned in the dialogic specification description (see 4.12), this protocol allows to send an *Inform* message to the other agents playing the same role as the sender, the sender itself being excluded.

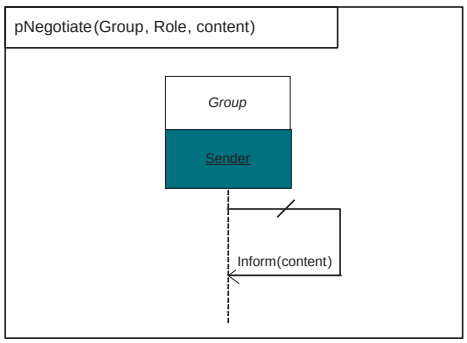


Figure 5.6: dLCGA Dialogic Specification: the pNegotiate Protocol

5.3.2.4 Normative Specification

Table 5.4 presents the normative description of the dLCGA using MAS4EVO. We will provide a textual description for the norms which differ from the ones defined for the LCGA (see 4.6.2.4), which are N11 and N12.

N01, N02, N03, N04, N05, N06, N07, N08, N09 and N10 are similar to the same norms of LCGA. N13 for the dLCGA is similar to N11 for the LCGA, and the same applies for norms N14 similar to N12 and N15 similar to N13.

			$\mathcal{SS}$	$\mathcal{FS}$		
n	φ	op	$bearer$	m	p	s
N01	true	obl	EvoBuilder	m1	—	Fabric
N02	true	obl	EvoAgent	m2	—	Fabric
N03	true	obl	EvoBuilder, EvoMember, Observer	m3	—	Fabric
N04	true	obl	EvoMember	m4	—	Fabric
N05	term==iterations	obl	EvoBuilder, EvoMember	m5	—	Fabric
N06	term==time	obl	EvoBuilder, EvoMember	m6	n	Fabric
N07	true	obl	Producer, Consumer	m7	—	Optimization
N08	true	obl	Consumer	m8	—	Optimization
N09	true	obl	Consumer, Producer	m9	—	Optimization
N10	true	obl	Producer, Observer	m10	—	Optimization
N11	true	obl	Solver	m11	is true ? (Reorganize)	Reorganization
N12	true	obl	Solver	m12	—	Reorganization
N13	true	obl	Observer	m11	—	Observation
N14	mode==batch	obl	Observer	m12	—	Observation
N15	mode==graphical	obl	Observer	m13	—	Observation

Table 5.4: Normative Specification of the dLCGA

The norm N11 is an obligation for the Solver to monitor if the condition for the reorganization is met.

The norm N12 is an obligation for the Solvers to fulfil the mission m12 which consists in the interactional goal pNegotiate(LCGA, Solver, groupNumber)), Optimization Scheme). This means that the Solvers have to negotiate their new SolvingUnit group number.

5.4 Conclusion

This chapter has presented two new competitive coevolutionary genetic algorithms which have been developed in DAFO, hLCGA and dLCGA, respectively hybrid and dynamic variants of the LCGA.

Their detailed model using MAS4EVO has been introduced, demonstrating the capacity of this new organizational model to describe such new coevolutionary genetic algorithms.

Indeed, for the hLCGA it was shown that by adding a new group and a new role in the SS and a a few new goals in the FS to the LCGA's (and adapting the NS accordingly) it is possible to create a new hybrid variant.

Similarly, by keeping the same SS as the LCGA and adding a reorganization scheme in its FS it is possible to create a new dynamic variant of LCGA.

Chapter 6

Implementation

Contents

6.1 Introduction

We have presented in Chapter 4 the MAS4EVO model and its utilization for modeling two existing CGAs, i.e. CCGA and LCGA. In chapter 5 we have introduced two new CGAs, a hybrid one (hLCGA) and a dynamic one (dLCGA), and their respective models using MAS4EVO.

After presenting those models, we now provide a description of the implementation of MAS4EVO that we call DAFO, Distributed Agent Framework for Optimization.

In this chapter, we first provide in section 6.2 an overview of DAFO's modular architecture. Then in the next sections, from 6.2.1 to 6.3, a detailed description of each component of DAFO is given, based on UML class diagrams. Finally section 6.1 presents DAFODL, the description language which allows the DAFO user to specify parameters concerning the agents organization and the algorithm's parameters.

6.2 DAFO Agent Architecture

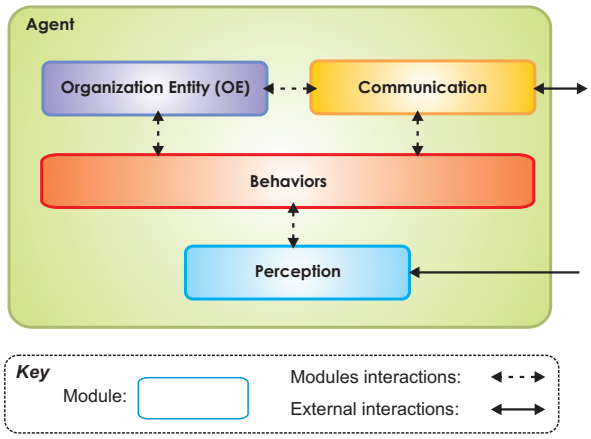


Figure 6.1: DAFO's Modular Architecture

Figure 6.1 provides a modular view of the DAFO agents architecture. These modules are the implementation of the conceptual model of the MAS4EVO agent presented in Figure 4.2. The implementation of the DAFO agents is based on the Madkit API provided by the Madkit platform.

The **Agents** module represents the three different agents, i.e. the Problem Solving Agent (PSA), the Observation Agent (OA) and the Fabric Agent (FA). This module embeds an internal engine which manages the lifecycle of the agent and the activation of its different internal modules.

The **Organization Entity (OE)** module represents the instantiation of the four MAS4EVO organization specifications (structural, functional, dialogic and normative). This module benefits from the agent API (Madkit) by using the available structural primitives, i.e. groups and roles.

The **Communication** module implements the interaction mechanisms and more particularly the interaction protocols defined in the MAS4EVO's Dialogic Specification. This module also benefits from the Madkit API since by using its message primitives.

The **Behaviors** module contains the set of behaviors, in a Jade-like way [157], available to the agents. Depending on the missions it has to fulfil, an agent will activate one or several behaviors. A behavior uses some skills to realize the goals corresponding to the mission.
The behaviors module is linked to the OE module since, it will be in charge of adopting role(s) in group(s) in the instantiation of the Structural Specification (SS) as well as satisfying the missions and goals defined in the instantiation of the Functional Specification (FS) which is part of the OE.
The behaviors module is also linked to the Communication module since some missions can imply interactional goals. In that case, interaction protocols embedded in the Communication module will have to be used.

The **Perception** allows the agent to perceive its environment, i.e. the other agents of the system and the MAS environment (which is the optimization problem provided by the user). The perception of the other agents is a functionality provided by the Madkit API.

The DAFO framework has thus been built on top of Madkit, which is implemented in JAVA. We consequently used the same programming language. In order to describe the object oriented implementation of DAFO, we make use of the Unified Modeling Language (UML). The following sections provide a detailed description of each of the aforementioned modules.

6.2.1 Agent Architecture

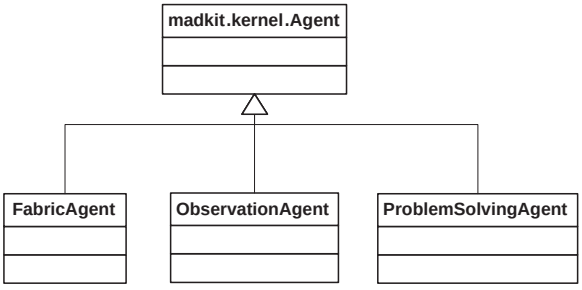


Figure 6.2: UML class diagram of DAFO's agents

Since DAFO is implemented on top of the Madkit API, all the agents of the system inherit from the Madkit *Agent* class, as represented in Figure 6.2. The three different agent types defined in MAS4EVO in 6.2.1, Observation Agent, Fabric Agent and Problem Solving Agent, are therefore modeled as three classes inheriting from the madkit.kernel.Agent class.

6.2.2 Agents' Behaviors

Behaviors are used by the agents, depending on the parameters set by the user, to ensure the satisfaction of the norms defined in the NS. Behaviors will therefore be in charge of realizing the missions defined in the Functional Specification and consequently the goals of these missions.

This is represented in the UML class diagram of Figure 6.3 in which the *Behavior* class has a link with the *Norm*. The *Norm* class has a link with the *Mission* class to express on what the norm is applied. A Mission contains one or several Goals (composition link). The *Plan* class represents the subgoals specifying another goal. This class is composed of the *Goal* class representing the subgoals. It has also a link with the same *Goal* class to define the root goal of the plan.

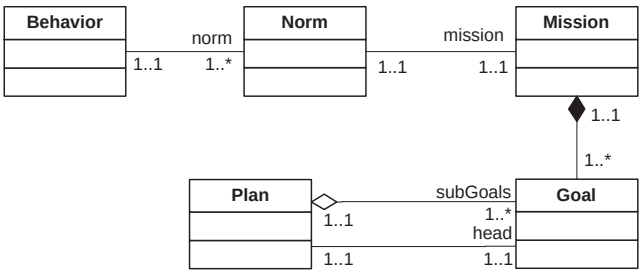


Figure 6.3: UML Representation of Norms

6.2.2.1 Problem Solving Behaviors

The *Problem Solving Agent* has the possibility to activate one or two behaviors among the *SGABehavior*, *CCGABehavior*,*LCGABehavior*, *DynamicBehavior*, *HybridBehavior* and *LocalSearchBehavior*. The corresponding UML class diagram is presented in Figure 6.4.

The possible combinations of two behaviors are:
- LCGABehavior + HybridBehavior: to model a dLCGA algorithm
- LCGABehavior + DynamicBehavior: to model a hLCGA algorithm

The *SGABehavior* class embeds the skills allowing a PSA to run a SGA. These are *createComputingGroups* and *joinComputingGroups* to fulfill the gCreateSubGroup(SGA,DAFO), gAdoptRole(EvoMember,DAFO) and gAdoptRole(Solver,SGA) organizational goals. It also possesses the *evaluate*, *genGA* and *sendSolutionsToObserver* skills to fulfil the *gRunGA* and *pInform(best)* goals of

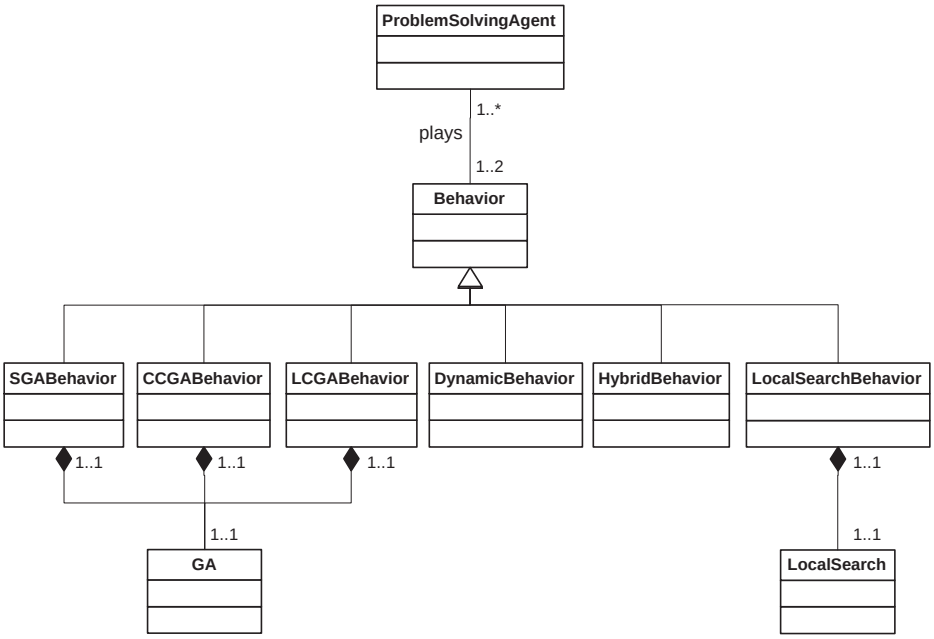


Figure 6.4: UML Representation of PSA Behaviors

the *Optimization scheme* (see SGA Functional Specification in Figure 4.10).

The *CCGABehavior* class embeds the skills allowing PSAs to run a CCGA. These are *createComputingGroups* and *joinComputingGroups* to fulfill the gCreateSubGroup(CCGA,DAFO), gAdoptRole(EvoMember,DAFO) and gAdoptRole(Producer,CCGA) organizational goals of the Fabric Scheme. It also possesses the *isReoganizationNeeded*, *joinComputingGroups*, *evaluate*, *genGA*, *sendToPSA*, *sendSolutionsToObserver* skills to fulfil the goals of the *Optimization scheme* (see CCGA Functional Specification in Figure 4.14).

The *LCGABehavior* class embeds the skills allowing PSAs to run a LCGA. These are *createComputingGroups* and *joinComputingGroups* to fulfill the gCreateSubGroup(LCGA,DAFO), gCreateSubGroup(SolvingUnit,LCGA), gAdoptRole(EvoMember,DAFO), gAdoptRole(Solver,LCGA), gAdoptRole(Consumer,SolvingUnit) and gAdoptRole(Producer,SolvingUnit) organizational goals of the Fabric Scheme (see LCGA FS in Figure 4.18). It also possesses the *evaluate*, *genGA* and *sendSolutionsToObserver* skills to fulfil the goals of the *Optimization scheme*.

The *DynamicBehavior* has to be combined with the *LCGABehavior* to provide the necessary skills to the PSA to build a dLCGA. Those additional skills are *isReoganizationNeeded*, *leaveComputingGroups* and *findNewComputingGroups* which allow to fulfil the goals of the Reorganization Scheme (see dLCGA Functional Specification in Figure 5.5).

The *HybridBehavior* must be combined with the *LCGABehavior* to fulfil the additional goals necessary to the PSA playing the *Solver* role to communicate with the PSA playing the *LocalSolver* role in case of a hLCGA (see hLCGA Structural Specification in Figure 5.2). These additional goals are *pInform* interactional goals.

The *LocalSearchBehavior* embeds the skills necessary to the PSAs playing the *LocalSearcher* role in case of a hLCGA. These skills are LS and sendToPSA which allow them to fulfill the functional goal gRunLS and the two interactional goals pInform(alpha) (see hLCGA Functional Specification in Figure 5.2.2.2).

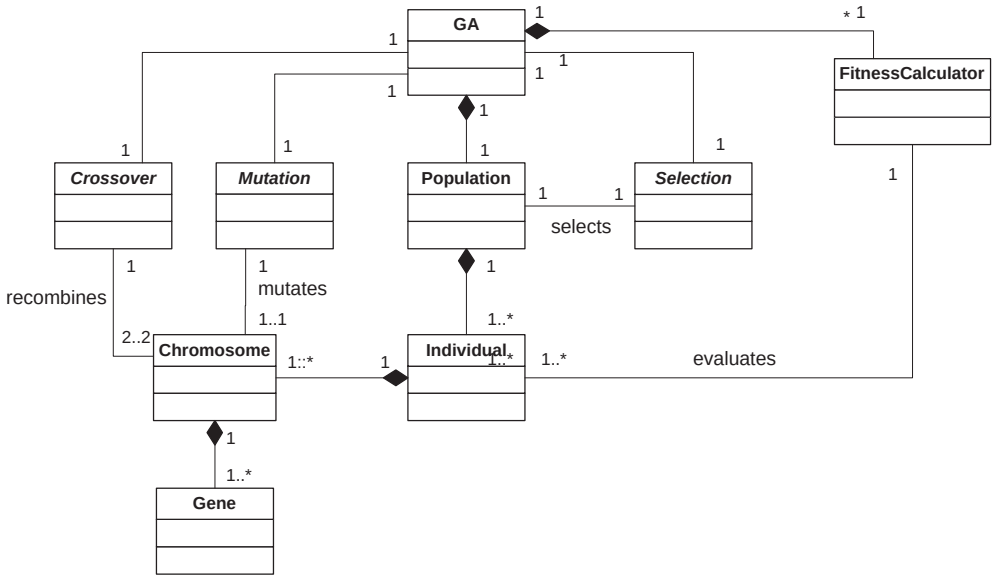


Figure 6.5: UML Representation of EvoAgents' Genetic Algorithms

The three behaviors *SGABehavior*, *CCGABehavior* and *LCGABehavior* possess a GA (see composition link in the UML class diagram of Figure 6.5). A GA has a *Population* of *Individuals* which contain one *Chromosome* composed of at least one *Gene*. The GA has access to different selections through abstract methods in the *Selection* class. The same way, the GA can apply different recombination and mutation operators through abstract methods in the *Crossover* and *Mutation* classes. The GA evaluates its individuals using a *FitnessFunction* class which is provided by the framework user.

The *LocalSearchBehavior* behavior possesses a local search algorithm as shown in the UML class diagram in Figure 6.6. This local search algorithm can be a Steepest Ascent Hill Climbing (SAHC), a Nearest Ascent Bit Climbing (NAHC), a Random Bit Climbing (RBC), a Dynamic Hill Climbing (DHC) or a Tabu Search. A local search algorithm is applied to one or several *Individuals*. To evaluate the individuals, the *LocalSearch* uses the *FitnessCalculator* class provided by the framework user.

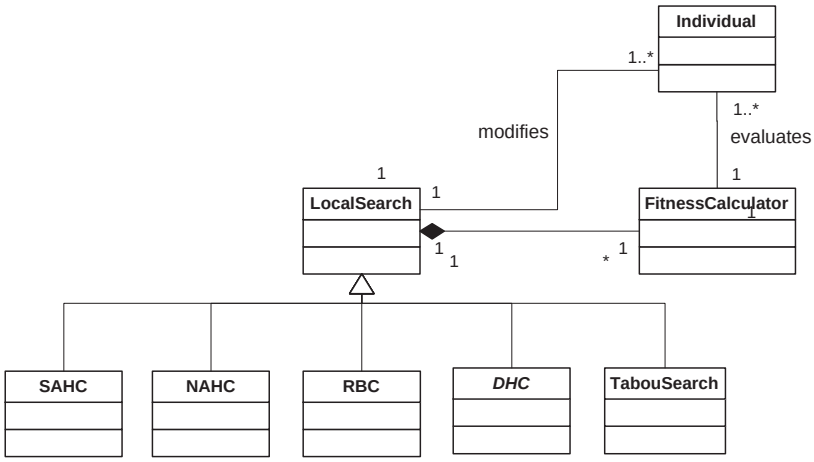


Figure 6.6: UML Representation of LSAgents' Local Search Algorithms

6.2.2.2 Observation Behaviors

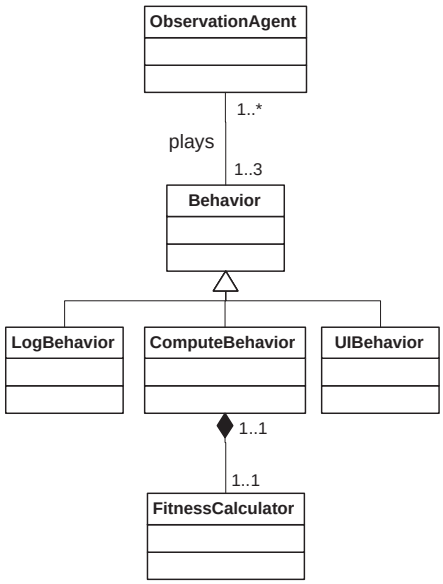


Figure 6.7: UML Representation of OA Behaviors

The *Observation Agent* has the possibility to use from one to three behaviors which are *LogBehavior*, *ComputeBehavior* and *UIBehavior*. The corresponding UML class diagram is presented in Figure 6.7.

The *LogBehavior* class embeds the skills allowing to log the results of the computation, *logBestInGenerations*, *computeAverage* and *logAveragedBestInGeneration*. These skills will be used to fulfil the functional goal *gLog*.

The *UIBehavior* will be used by the Observation Agent in addition to the *LogBehavior* if DAFO is used in graphical mode. It contains the skills to draw the graphs corresponding to the logged results, *drawBestInGeneration* and *drawAveragedBestInGeneration*. These skills will be used so as to fulfill the *gGraphAndLog* functional goal.

The *ComputeBehavior* class embeds the skills allowing to compute the global solution (if required by the optimization algorithm), *aggregateReceivedSolutions* and *evaluateGlobalSolution*. In order to evaluate the solution on the optimization problem, the *ComputeBehavior* class possesses a *FitnessCalculator* (composition link) provided by the user. These skills will be used to fulfil the functional goal *gGlobalFitness*.

6.2.2.3 Fabric Behaviors

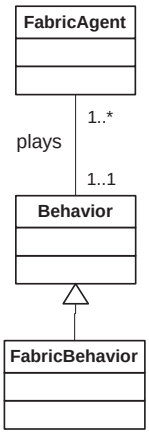


Figure 6.8: UML Representation of FA Behaviors

The *Fabric Agent* has only one behavior which is the *FabricBehavior*. The corresponding UML class diagram is presented in Figure 6.8.

This behavior thus possesses all the Fabric Agent skills, the first one being *parseEvoParameters* to parse the parameters set by the user (details of these parameters in section 6.4). It also embeds the *createPSAs* skill used to fulfill the *gInstantiate(a)* organizational goal(s).

It finally has the *sendParametersToPSA*, *sendParametersToObservation*, *isComputationalTimeFinished*, *sendStartToPSA* and *sendStopToPSA* skills which will use either the *pInform* or the *pTime* protocols (presented in section 6.2.4) to fulfill the corresponding interactional goals.

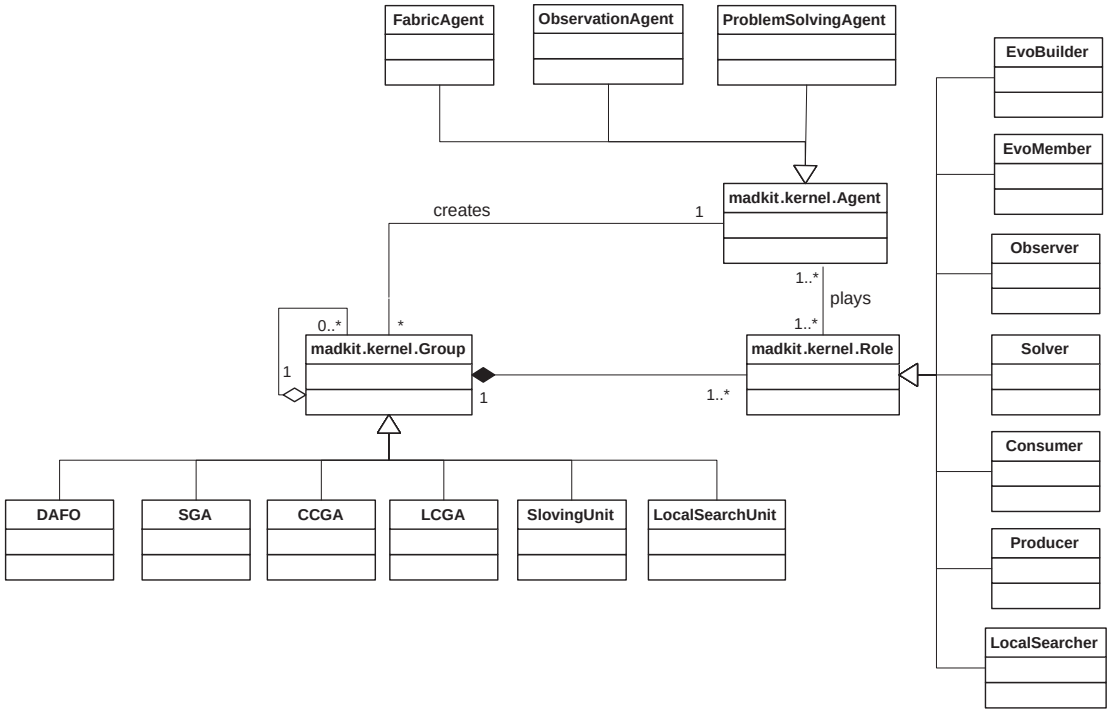


Figure 6.9: UML class diagram of DAFO's organization entity

6.2.3 Agent Organization Management Module

As previously mentioned, the implementation of the OE benefits from the available structural specification of the Madkit API. As represented in Figure 6.9, Madkit agents can play one or several roles, a role being played in one group. A group can contain other group(s) in order to have subgroups and a Madkit Agent can create groups.

The set of roles defined in MAS4EVO is therefore implemented in DAFO as classes inheriting from the *madkit.kernel.Role* class. These are the *EvoBuilder*, *EvoMember*, *Observer*, *Solver*, *Consumer*, *Producer* and *LocalSearcher* classes.

The same way, the different groups, *DAFO*, *SGA*, *CCGA*, *LCGA*, *SolvingUnit* and *LocalSearchUnit* inherit from the *madkit.kernel.Group* class of the Madkit API.

6.2.4 Agent Communication Module

The communication between agents is based on the message API of Madkit. The three different interaction protocols specified in MAS4EVO are modeled as three different classes, *pTime*, *pInform* and *pNegociate*, which inherit from a *Protocol* class (see Figure 6.10.

A *Protocol* class is composed of at least one message, it thus has a composition link with the

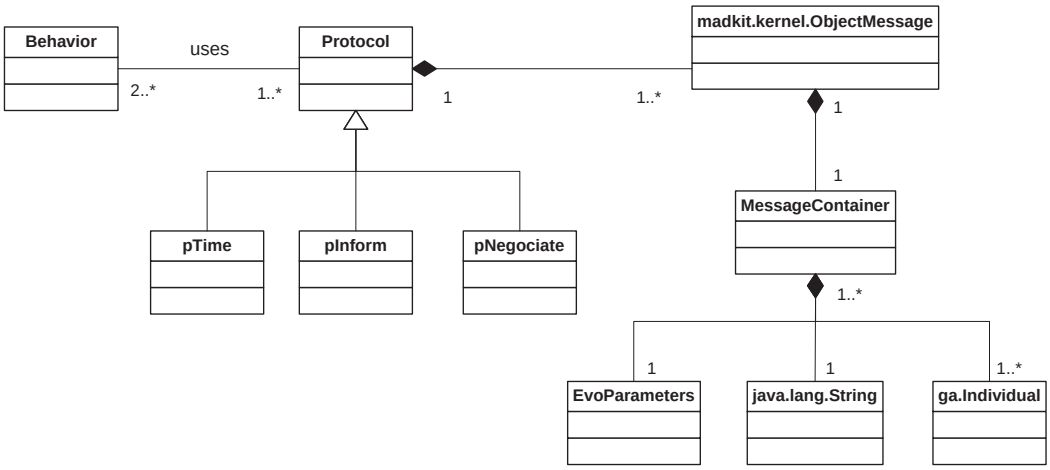


Figure 6.10: UML class diagram of DAFO's communication

ObjectMessage class provided by Madkit. The content of a message is encapsulated in a MessageContainer (composition link) which can contain one instance of *EvoParameters*, on instance of a *String* or one to several instances of *Individuals*.

A protocol is used by at least two roles, that is why a link exists between the *Protocol* and the *Behavior* classes.

6.2.5 Agent Perception Module

The perception of the agent's environment is partially provided by the Madkit API. Indeed, Madkit permits to its agents to have a perception of the other agents. The perception module thus adds the possibility to the agents to perceive the other component of the environment: the optimization problem. Agents have a perception of the optimization problem provided by the user through the *FitnessCalculator* class. This class can be used by the *ProblemSolvingAgent* in the *SGABehavior*, *CCGABehavior* and *LCGABehavior* through the GA, by the LocalSearchBehavior through the LocalSearch (see Figure 6.4) and finally by the ObservationAgent in the *ComputeBehavior*.

6.3 Agent Platform

In the previous section we already mentioned how DAFO takes benefit from the Madkit API in concerning the implementation of the agents, organization structure and communication.

We consequently used the Madkit platform [29] on which those agents are executed. The choice of Madkit was additionally motivated by its distribution capabilities and its performance compared to other agent platforms. For more details, a survey of multi-agent platforms is provided in Appendix A.

DAFO's distribution therefore relies on Madkit. Indeed, the MadKit platform provides a specific agent, called *Communicator Agent*, which brings distribution capabilities and allows different Madkit kernels to form a network from which agents may communicate transparently. This *Communicator Agent* provides distributed message passing through socket connections. These connections are established by using the host name and optionally the port on which the distant communicator is listening. It also handles organizational information synchronization on multiple distant agent kernel. Groups and roles created on a platform in distributed mode are seen from the other connected platforms. Once the connection is made between two kernels, all organization tables are synchronized, thus groups (and their respective roles and agents) which have been created in distributed mode are automatically seen by both kernels, assuming that the two kernels are connected to the communities where these groups belong to. When one kernel wants to connect to an already formed network, i.e. several kernels connected together, it is just necessary to connect to one of them since all subsequent connections between this kernel and the other ones will be done automatically (as illustrated in Figure 6.11.

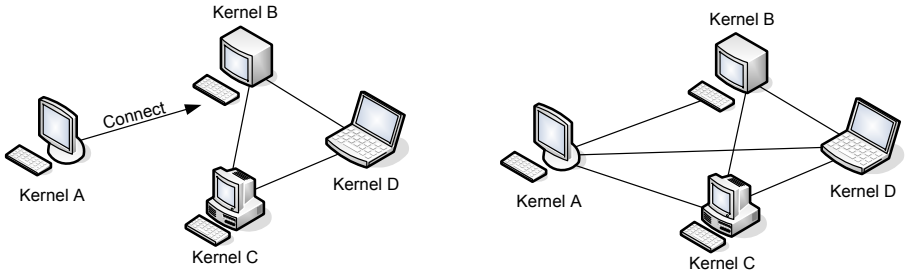


Figure 6.11: Madkit kernels connection

6.4 DAFODL: DAFO Description Language

The objective of DAFODL (DAFO Descritpion Language) is to provide a language which is simple enough to be understood by the framework user but also sufficiently structured to be interpreted by the program. The optimization of a problem using one algorithm modeled with MAS4EVO and implemented in DAFO requires the definition of a configuration file based on DAFODL which will specify on the one hand the organization of agents as presented in chapters 4 and 5 and on the other hand parameters related to CGAs.

DAFODL is an XML configuration file based on a DTD provided in Appendix B. This file is structured in two main parts, the first one describing the agents organization and the second one describing the genetic algorithm parameters. And optional third part describes the local search parameters in case of a hybrid algorithm.

Table 6.1 presents all the usable tags in DAFODL and the possible values which can be used for each of them.

	Tag	Possible Values	Description
Org.	topology	completegraph, ring, simplelist	
	numberofneighbors	1 ⩽ integer ⩽ numberofagents-1	Only valid in case of a ring topology
	numberofagents	1 ⩽ integer ⩽ n	Number of evolutionary agents
	reorgstep	1 ⩽ integer ⩽ n	Number of generations between two reorganizations
GA Params	algorithm	SGA, CCGA-1, CCGA-2, LCGA, dLCGA, hLCGA	Algorithm type
	optimization	maximize, minimize	Minimization or maximization problem
	fitnessclass	String	Name of the fitness evaluation Java class
	experiments	1 ⩽ integer ⩽ n	Number of experiments
	terminationcondition	generation, functionevaluation, time	Type of termination condition
	terminationconditionvalue	1 ⩽ integer ⩽ t	Value of the termination condition
	numchroms	1 ⩽ integer ⩽ n	Number of chromosomes in the population
	numgenes	1 ⩽ integer ⩽ n	Number of genes in a chromosome
	sizegenes	1 ⩽ integer ⩽ n	Number of bits in a gene
	crossover	1Point, 2Point, Uniform	Crossover operator
	crossrate	0 ⩽ float ⩽ n	Crossover rate
	mutrate	0 ⩽ float ⩽ 1	Mutation rate
	elitenumber	0 ⩽ float ⩽ numberofagents	Number of best indiv. kept unchanged in next generation
LS Params	lsalgorithm	NAHC, SAHC, DHC, RBC, TS	Local search algorithm
	lsexchangedinformation	best, poprate	Information sent from EvoAgent to LSAgent
	lspopulationrate	0 ⩽ float ⩽ 1	Used if poprate exchangedinformation
	lsterminationcondition	restricted, complete	Local search termination condition

Table 6.1: DAFODL Structure

The following provides a description of those tags available in DAFODL.

Organization Parameters:

The *topology* has to be completegraph for the CCGA while it can also be a simple list or a ring with the LCGA. When using the ring topology, it is possible to augment the number of neighbors of all the nodes in the ring using the *numberofneighbors* parameter as represented in Figure 4.16. The *numberofagents* parameter sets the number of evolutionary agents that will be instantiated and at the same time the number of local search agents if a hybrid algorithm is set. The *reorgstep* represents the number of generations each evolutionary agent will execute between two reorganizations.

GA Parameters:

The *algorithm* parameter specifies the GA that has to be instantiated. *optimization* specifies if the optimization problem is a maximization or minimization one. The *fitnessclass* provides the name of the Java class implementing the *FitnessCalculator* class of DAFO and implemented by the user to

evaluate the individuals on its optimization problem. *experiments* specifies the number of runs to execute, *terminationcondition* allows to choose between three stopping criterion which can be a number of generations, a number of fitness function evaluation or a computational time. The value of the chosen termination condition is provided in *terminationconditionvalue* (given in milliseconds for the time). *numchroms* gives the population size in each evolutionary agent, *numgenes* gives the number of genes in a chromosome and *sizegenes* gives the number of bits in a gene (i.e. only a binary representation is available). The operators are set with *crossover* to choose between a 1-point, a 2-point or a uniform crossover, *crossrate* to specify the crossover rate (between 0 and 1), *mutrate* to set the mutation rate (between 0 and 1) and *elitenumber* which specifies the number of best individuals which are kept unchanged for the next generation.

```
<?xml version='1.0' encoding='UTF-8'?>
<evoframework>
    <organisation>
        <topology>Ring</topology>
        <numberofagents>10</numberofagents>
    </organisation>
    <geneticparameters>
        <algorithm>hLCGA</algorithm>
        <fitnessclass>MadhocFitnessCalculator</fitnessclass>
        <experiments>20</experiments>
        <terminationcondition>Time</terminationcondition>
        <terminationconditionvalue>60000</terminationconditionvalue>
        <numchroms>100</numchroms>
        <numgenes>2</numgenes>
        <sizegenes>16</sizegenes>
        <crossRate>0.8</crossRate>
        <mutrate>0.03</mutrate>
        <elitenumber>1</elitenumber>
    </geneticparameters>
    <localsearchparameters>
        <lsalgorithm>TabuSearch</lsalgorithm>
        <lsexchangedinformation>PopulationRate</lsexchangedinformation>
        <lspopulationrate>0.01</lspopulationrate>
        <lsterminationCondition>Restricted</lsterminationCondition>
    </localsearchparameters>
</evoframework>
```

Table 6.2: Example of DAFODL based configuration file

Local Search Parameters:

Those parameters are optional since they are required only in case of a hLCGA. *lsalgorithm* allow to choose between five different local search algorithms (described in 5.2). Two different individuals exchange strategies between the evolutionary agent and the local search agent can be selected in *lsexchangedinformation*, either the current best individual of the evolutionary agent or a population rate. In case of a population rate, its value has to be set in *lspopulationrate* (between 0 and 1). Finally it is possible to choose between two termination conditions for the local search algorithm in

lsterminationcondition, restricted (stops as soon as an improved individual is found) or complete (fully executed).

Table 6.2 provides an example of a configuration file based on DAFODL for a hLCGA with a tabu search algorithm.

This description will be parsed by the *Fabric Agent* in order to instantiate the corresponding *Evolutionary Agents* and *Local Search Agents* organization as presented in Figure 4.3. Afterwards, the different parameters related to the genetic and local search algorithms will be transmitted to these agents so that they can start their computation.

6.5 Conclusion

After the presentation of the MAS4EVO model and its usage to model different CGAs in chapters 4 and 5, this chapter has provided details concerning the implementation of these MAS4EVO models in the DAFO framework.

This chapter first brought in section 6.2 a general view of DAFO's agent architecture based on five components which are:
- an agent component,
- an organization entity component,
- a communication component,
- a behavior component,
- a perception component.

Afterwards, a detailed description of each of these components has been given, based on UML class diagrams. The agent module possesses an inference engine which manages the lifecycle of the agent and the activation of its different internal modules (see section 6.2.1. The organization entity provides structural specifications as defined in the SS (see section 6.2.3). The communication component provides the interaction protocols defined in the Dialogic Specification of MAS4EVO (see section 6.2.4). The behavior component provides the skills necessary to fulfill the goals and missions defined in the FS while ensuring the respect of the norms defined in the NS (see section 6.2.2). Finally the perception module adds the possibility to the agents to perceive the other agents and the other component of the environment, i.e. the optimization problem (see section 6.2.5).

In section 6.3 we detailed how the usage of the Madkit platform facilitates the distribution of the agents and therefore of the CGAs.

Finally, in section 6.4, we introduced an xml-based language called DAFODL (DAFO Description language) which allows the user to define the algorithm he wants to use on his optimization problem.

To this end, DAFODL allows to describe the necessary organizational, genetic and local search parameters. An example of DAFODL usage for a hLCGA was shown.

In the following two chapters we now introduce two application scenarios of the DAFO framework on business optimization problems. On the one hand this will allow to validate the DAFO framework and its organizational representation of CGAs and on the other hand this will permit to evaluate the new LCGAs (hybrid and dynamic) we introduced in Chapter 5.

Part III

Experimentations

Chapter 7

Static Problem Case Study: Inventory Management

Contents

In the previous chapter, we have introduced the organizational and reorganizational multi-agent model and the implementation of our framework dedicated to distributed coevolutionary optimization, DAFO.

In the coming chapter, we present the first business optimization problem we have tackled using DAFO which is a stock management problem called ICP (Inventory Control Parameter) problem. This problem was previously studied by Ericksson and Olsson in [14] who compared the performance of different CCGA variants (CCGA-1 and CCGA-2) to a generational GA. We have extended their analysis by comparing the CCGA to the LCGA (which induced some decomposition issues) and two new variants we have created, a hybrid LCGA (hLCGA) and a dynamic LCGA (dLCGA).

We start by describing the ICP problem, then we provide some experimental results obtained comparing LCGA, including the problem decomposition we chose, and CCGA. In section 7.5 we give a detailed description of the hybridization of LCGA, we validate it on a classical test function (Rosenbrock) and we analyze its results on the ICP problem. Finally section 7.6 introduces the dynamic variant of LCGA, dLCGA, and its experimental results on the ICP problem are studied.

7.1 Problem Description

Inventory control is defined by the activities, techniques, and methods for maintaining an accurate stock of items at a desired level. It may concern finished goods, parts, work in progress, or raw materials. Inventory control has to deal with conflicting goals of finance, production and marketing. It seeks to find the best balance between these goals with the objective of minimizing the total cost.

As a real case study, we tested and compared our different algorithms, LCGA, hLCGA and dLCGA, on the ICP (Inventory Control Parameter) problem. A complete description of this problem can be found in [14], where a classical Genetic Algorithm is compared to four variants of CCGA (Cooperative Coevolutionary Genetic Algorithm) approaches (CCGA-1 and CCGA-2 both used with two different representations). ICP objective consists in defining the couple OP/OQ, order point - order quantity (when and how much to order) for each stock item. Whenever it reaches the order point, which is composed of an expected demand during lead time plus a safety stock, an order is released for a fixed order quantity (see fig. 7.1). The parameters OP/OQ are considered as fixed. The OP-OQ model is used for controlling warehouses of independent demand items (as opposed to dependent ones), which means that external factors induces variation in demand for this kind of items. These external factors induce random variations in the demand for such items, therefore independent demand must be forecasted.

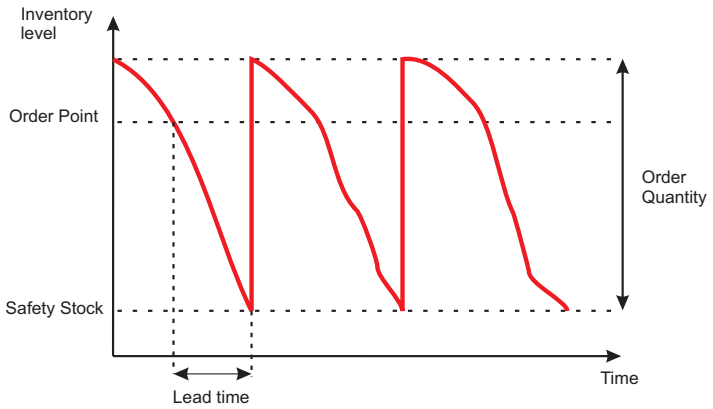


Figure 7.1: ICP Optimization Problem

The fitness evaluation is achieved through the run of a simulation where a set of customer purchase transactions are processed to calculate the total cost resulting from inventory decisions (see Figure 7.2).

This total cost is the sum of different costs :

- lost sales costs : depending on customer delivery delay (time that a customer has to wait due to inventory shortage)
- transportation costs : depending on the number of replenishment and their size
- order costs : depending on the number of orders
- storage space costs : depending on the size of the warehouse (belongs to the carrying costs)
- order costs : depending on the number of orders

In our experiments we have added some interdependencies between the items by adding a benefit for joint orders and joint replenishment. In other words, whenever multiple orders and/or replenishment occur at the same time (i.e. at the same timestamp), there is a proportional benefit.

To summarize the simulation, it is first initialized by collecting the order point and order quantity values for all items from the solution under evaluation. This is done by initializing the above costs to zero and by initializing the stock levels. During the simulation, the transaction file containing a number of customer orders is read incrementally and each customer order is handled in turn. A transaction is composed of a transaction number, a timestamp (that can be interpreted as a day), an item number and a quantity. Deliveries are made, stock levels updated and replenishment orders placed when stock levels fall below order points. For each customer order, replenishment order or delivery, the different cost sums are updated. The final step is to sum the different costs to obtain a total inventory cost from which a fitness value can be computed.

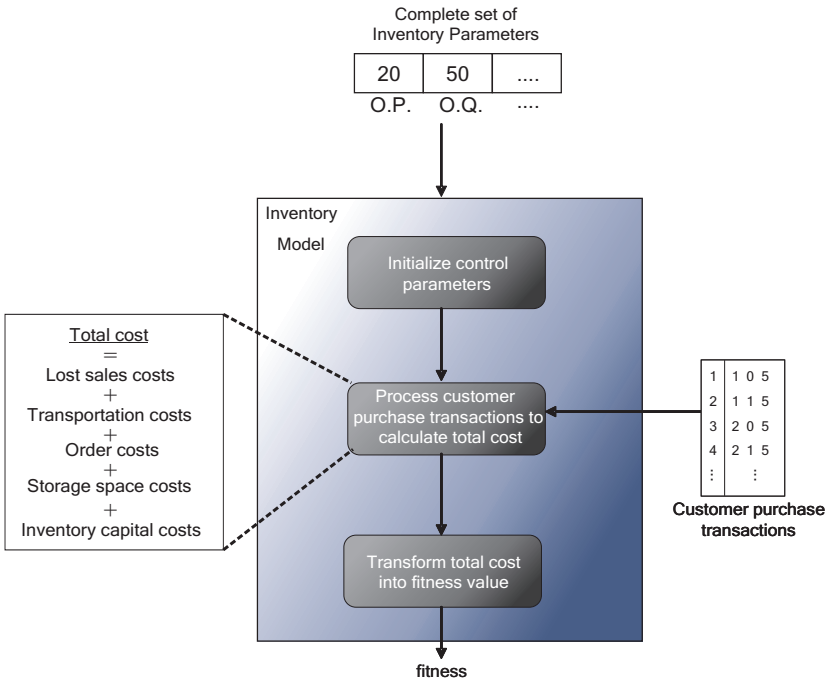


Figure 7.2: Fitness evaluation by inventory simulation

The following section presents a first extension to Ericksson's work by adding a comparison to another coevolutionary GA, the LCGA.

7.2 Solution Encoding

For each evolutionary methods studied here, a chromosome is represented as a binary string (16 bits per parameter/gene), as can be seen in Figure 7.3.

Each gene represents a single inventory control parameter. Each parameter is an integer within an interval for which the minimum and maximum value can be configured for every item individually. For instance, it is possible to have an order quantity ranging from 1 to 100 for one item and from 50 to 500 for another item. However, in the experiments of this dissertation, order point quantities always range from 4 to 99 and order quantities always range from 1 to 100.

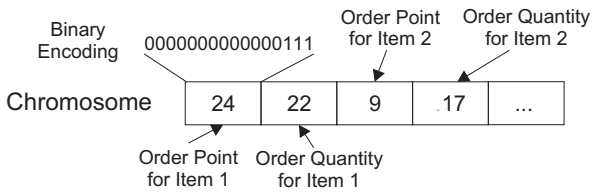


Figure 7.3: ICP Solution Encoding

Thus substrings of 7 bits would be sufficient to encode these intervals ($2^7 = 128$), but if these were mapped directly to integers some combinations would represent integers which do not belong to the interval. In order to avoid those illegal parameter values, the binary string is mapped to an integer number belonging to the interval of the parameter.

Like Eriksson we used 16 bits which is more than enough for coding the interval of integers and the unnecessary extra bits increase the search space with redundant solutions. However, one advantage of this "over-representation" is that it permits to change the interval of a parameter without changing the number of bits encoding it.

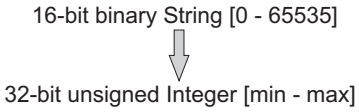


Figure 7.4: Mapping of binary string to integer number

7.3 Problem Decomposition

The representation used for the SGA is one individual representing all the parameters for all the items. For instance there are 10 items in the warehouse, for each item we have both order point and order quantity parameters, the chromosome will have 2*10 = 20 genes.

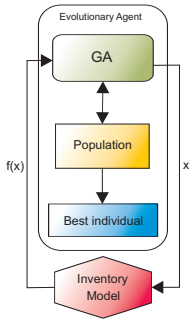


Figure 7.5: ICP Optimization using SGA

When using a coevolutionary GA like CCGA and LCGA, each subpopulation is in charge of optimizing one subset of the global solution.

In his work, Eriksson compared four different versions of CCGA. We have selected the one that provided the best results : CCGA-1R2 (CCGA with credit assignment 1 and representation 2). Credit assignment 1 means that the fitness of an individual is obtained by combining it with the current best

individual from each of the other (temporarily frozen) subpopulation. Contrary to representation 1, that fits the suggestions of Potter and De Jong, where an individual represents one parameter (order point or order quantity), in representation 2 (see Figure 7.6) an individual represents all parameters for an item (order point and order quantity). Representation 2 provided better results and thus confirmed the hypothesis that CCGA provides better results when there are weak dependencies among the parameters and equivalent results to the standard GA when strong dependencies occur. It is easily understandable that order point and order quantity are strongly dependent parameters.

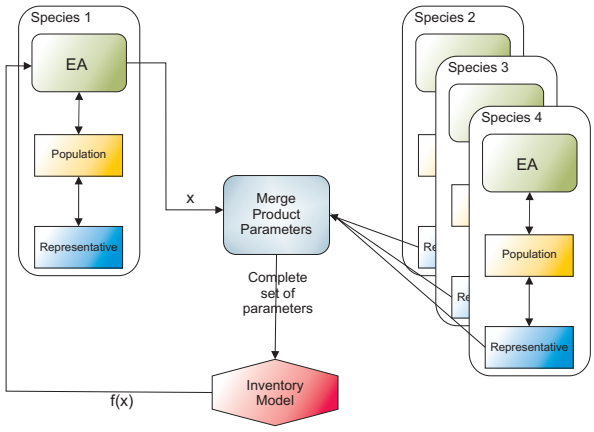


Figure 7.6: ICP optimization using CCGA with representation 2

The representation we used in LCGA and the new dynamic and hybrid variants (hLCGA and dLCGA) we introduce in this chapter is different from the one used for CCGA. Indeed, the version of LCGA we have implemented uses a ring topology (contrary to the fully connected topology of the CCGA) where each agent optimizes the order point and order quantity for one item but under the constraint of a single neighbor. To illustrate our solution, let us use a simple example with four agents and thus four different items in the stock (see fig. 7.7). Agent A_0 will evaluate its individuals using the individuals received from its neighbor A_3 by processing the transactions concerning its item (item 0) and its neighbors item (item 3) and the process is the same for the other agents in the ring. This way, evaluating an individual in each subpopulation in LCGA requires less transactions processing as opposed to CCGA in which each subpopulation has to process the whole transaction stream.

When all the agents have run once their subpopulation, the global solution (consisting of the best individuals of each agent) is evaluated on the whole transaction stream.

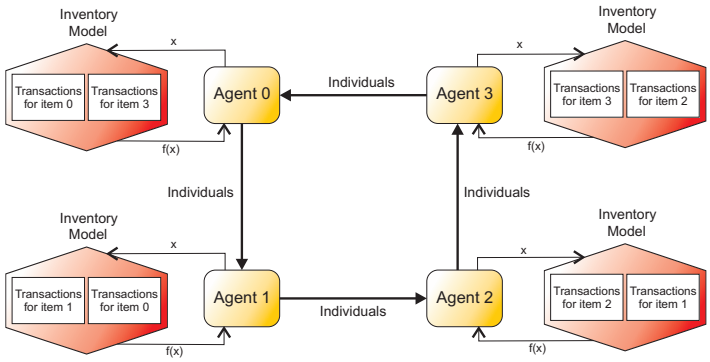


Figure 7.7: ICP optimization using LCGA

7.4 LCGA vs. CCGA

Fixing some constraints such as initial level, maximum level, lead time, we have compared the performance of a Simple GA (see fig. 7.5), a CCGA and LCGA on this ICP optimization problem.

7.4.1 Experimental Results

The following parameters were set for all the algorithms : population(s) size was equal to 100, p_k = 0.6 (crossover probability) and p_m = 0.05 (mutation probability). Each result presented hereafter is the average obtained on 30 independent runs.

Number of Subpopulations	3, 10, 100
(Sub)Population size	100 individuals
Termination Condition	10,000 (100,000 for 100 items) function evaluations
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.6
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual

Table 7.1: Parameters used for genGA, CCGA and LCGA

Fig. 7.8 shows the results obtained with the three algorithms for 3, 10 and 100 stock items and respectively 360, 1200 and 12000 transactions. It is clear that LCGA outperforms the SGA both in terms of speed of convergence and in the minimum cost found in the three problem instances.

Comparing LCGA to the CCGA on the smallest problem instance (3 items and 360 transactions), LCGA converges faster and reaches almost the same minimum cost (1292.49$ for LCGA and 1290.27$ for CCGA). We also have to take into account that the computation time for required by the LCGA is lower than for running the CCGA, given that local fitness function evaluations for LCGA are cheaper and require less agent interactions. Indeed, one subpopulation in LCGA only requires to process the

transactions of the item it optimizes the parameters and its neighbor's (thus 2 items) while in CCGA each subpopulation has to process the whole transaction stream.

ICP Parameters	GA	Result
3 Items, 360 Transactions	genGA	1334.90
	CCGA	1290.27
	LCGA	1292.49
10 Items, 1200 Transactions	genGA	4671.77
	CCGA	3896.02
	LCGA	4211.16
100 Items, 12000 Transactions	genGA	49453.70
	CCGA	42671.61
	LCGA	48537.69

Table 7.2: Results of SGA, CCGA and LCGA on the ICP problem with 3, 10 and 100 items and 320, 1200 and 12000 transactions

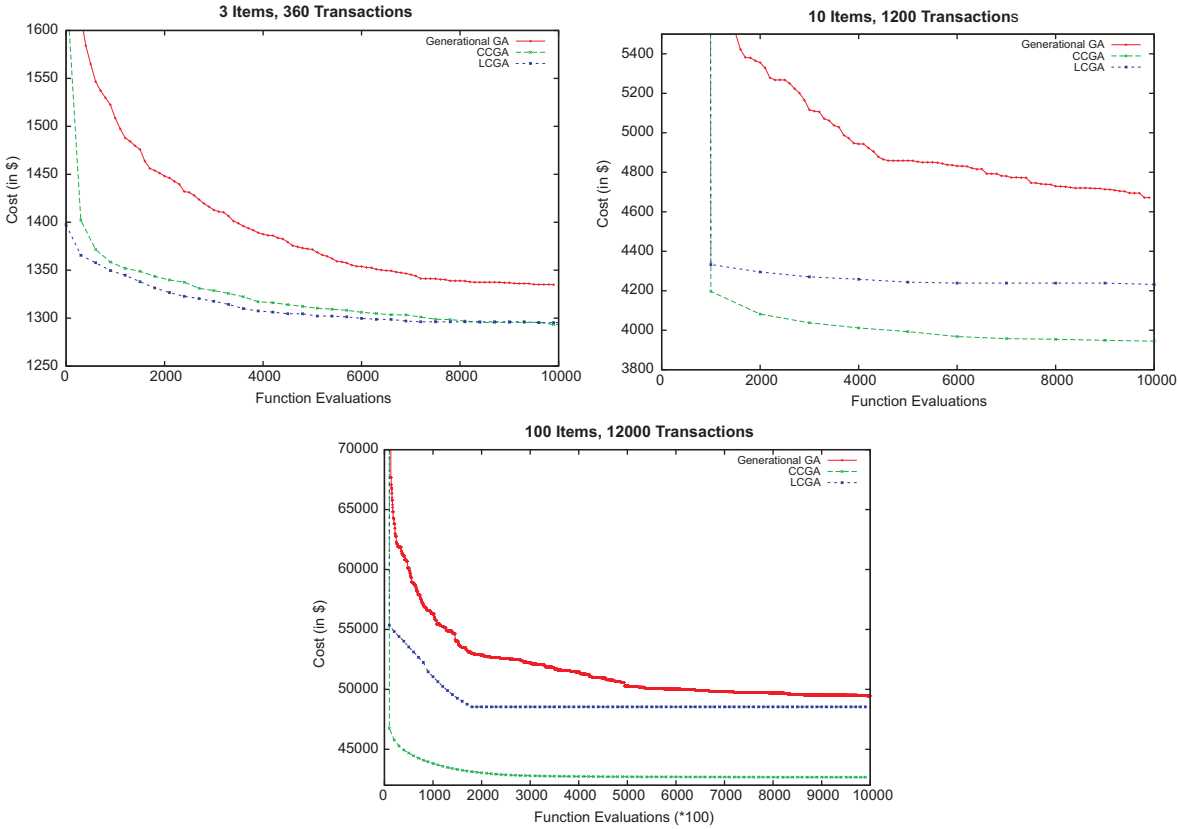


Figure 7.8: Average results on 30 runs for 2, 10 and 100 items using the the SGA, CCGA and LCGA

When dealing with bigger problem instances, i.e. 10 and 100 items, CCGA achieves better performance than LCGA, this difference increases with the problem complexity (see Figure 7.8). The LCGA

gets stuck in some local optimum after 1000 function evaluations in the 10 items instance and after 2000 in the 100 items instance. One reason for this loss of quality comes from the problem decomposition used for LCGA which implies a loss of information compared to CCGA. However, concerning the processing time, the difference has increased the opposite way, since the LCGA reaches a lower quality solution than CCGA but in a much shorter time. Indeed, each subpopulation in LCGA still processes the transactions of 2 items while in CCGA subpopulations have to process the transactions for respectively 10 and 100 items (i.e. 5 and 50 more transactions to process).

7.5 hLCGA

hLCGA, a new hybrid LCGA, has been described in chapter 5.2. In the following section we provide some experimental results, first on a classical test function (i.e. the Rosenbrock test function) and then on the ICP optimization problem.

7.5.1 Experimental Results

7.5.1.1 Validation of the hLCGA: the Rosenbrock test function optimization

We have experimented the different hLCGA variants and compared them to the "basic" LCGA on a classical function optimization problem known to be very hard: the Rosenbrock function that is part of De Jong's five function test suite [9] (see Figure 7.9). The Rosenbrock's function is a continuous and unimodal function :

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2\right); x \in R^n, \tag{7.1}$$

with $-2.12 \leq x_i \leq 2.12$, a global minimum $f_2(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 1, \ldots, 1)$. This global optimum is inside a long, narrow, parabolic shaped flat valley. Finding the valley is trivial, however converging to the global optimum is difficult.

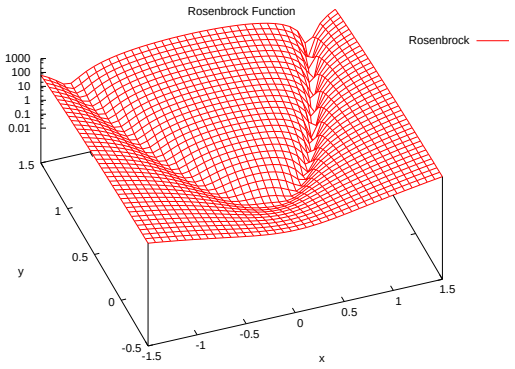


Figure 7.9: Rosenbrock Function graph for n=2

Using (h)LCGA, we consider the problem of minimizing the Rosenbrock's function as a problem of seeking a minimum in a distributed fashion. We use a multi-agent system, within our framework, with a game-theoretic model of interaction among agents as shown in the interaction graph represented in Figure 7.10.

For LCGA, each Problem Solving Agent (PSA), optimizes its locally defined function which depends

Figure 7.10: Rosenbrock LCGA Interaction Graph

only on its x_i and the $x_i + 1$ of its neighbor using a Simple GA (SGA). These agents are therefore referenced as "PSA GA" in Figure 7.10.

$$f_2^i(\mathbf{x_i}, \mathbf{x_{i+1}}) = 100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2); \tag{7.2}$$

This way, each PSA GA runs one subpopulation and individuals in one subpopulation code solution for a variable x_i. For hLCGA a PSAs running one of the previously mentioned local search algorithms, referenced as PSA LS, are added as illustrated in Fig. 7.11. In each generation the PSA GA will send a predefined percentage of its population to the PSA LS that will run one chosen local search algorithm and sends back the optimized individuals.

The performances of the LCGA and the five versions of hLCGA have been evaluated on one instance of the Rosenbrock problem of size n = 10. The following parameters were set for all the algorithms : sub-populations size was equal to 100, 16 bits binary representation, two-point crossover with $p_k = 0.8$ (crossover probability) and bit flip mutation with $p_m = 0.03$ (mutation probability).

Experiments have been conducted with the following local search parameters: *exchange rate* = 0.35 for hLCGA with SAHC, NAHC, RBC, DHC and 0.03 for Tabu Search. Another strategy exchanging only the best individual has also been tested. Each exchanged strategy (best individual and population rate) was experimented with both restricted and complete local search. All the results

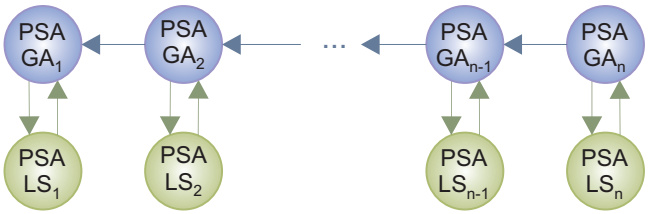


Figure 7.11: Rosenbrock hLCGA Interaction Graph

obtained are presented in Table 7.4.

Number of Subpopulations	10
SubPopulation size	100 individuals
Termination Condition	100 generations
Representation	16 bits binary
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual
Local search algorithms	SAHC, NAHC, RBC, DHC and TS
Local search termination condition	Complete search, Restricted search
Local search exchange strategy	best individual, population rate=0.35 (except 0.03 for TS)

Table 7.3: Parameters used for LCGA and hLCGA on the Rosenbrock function

The results presented in fig. 7.12 represent the averaged best of generation over 30 experiments for LCGA and hLCGA using populations rate exchange strategy combined with complete local search (with and without zoom).

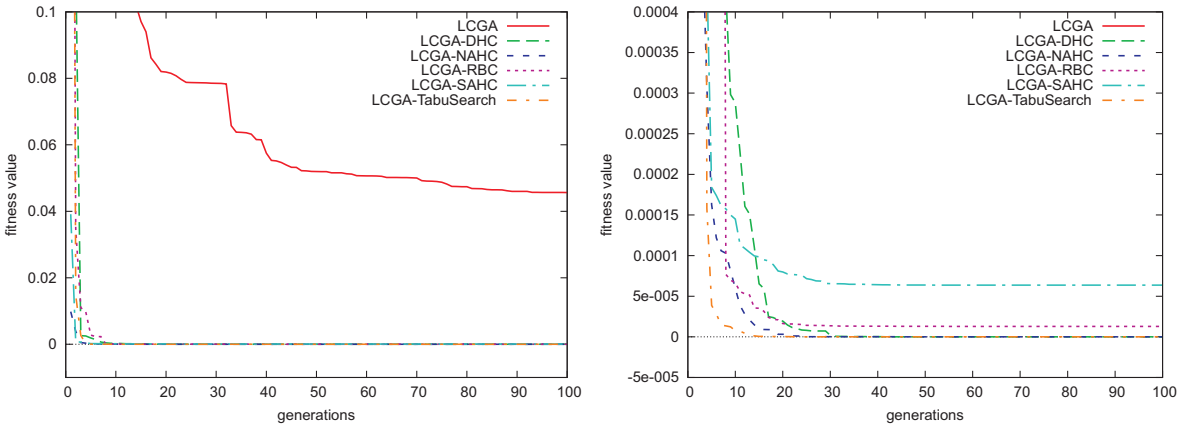


Figure 7.12: Rosenbrock n = 10, LCGA vs hLCGA with population rate exchange strategy and complete local search

It clearly appears that for such a continuous and unimodal problem all hLCGAs using populations rate exchange strategy combined with complete local search outperform LCGA both in terms of convergence speed and best result found, the overall best being LCGA-TS that converges the fastest to the global optimum (i.e. *f(x)*=0). LCGA-SAHC and LCGA-RBC are the only ones that do not converge to this global optimum. As expected, the drawback is the additional computational time required for the local search algorithms execution. Indeed, when it takes 3 seconds for LCGA to perform one experiment, it increases up to 27 seconds in the worst case for LCGA-TS. Taking this parameter into consideration, LCGA-RBC is the fastest with 4 seconds but as previously mentioned

it gets stuck in a local optimum, consequently LCGA-NAHC becomes the best choice since it also reaches the global optimum and takes 5 seconds for one experiment.

The results presented in fig. 7.13 represent the averaged best of generation over 30 experiments for LCGA and hLCGA using populations rate exchange strategy combined with restricted local search (with and without zoom).

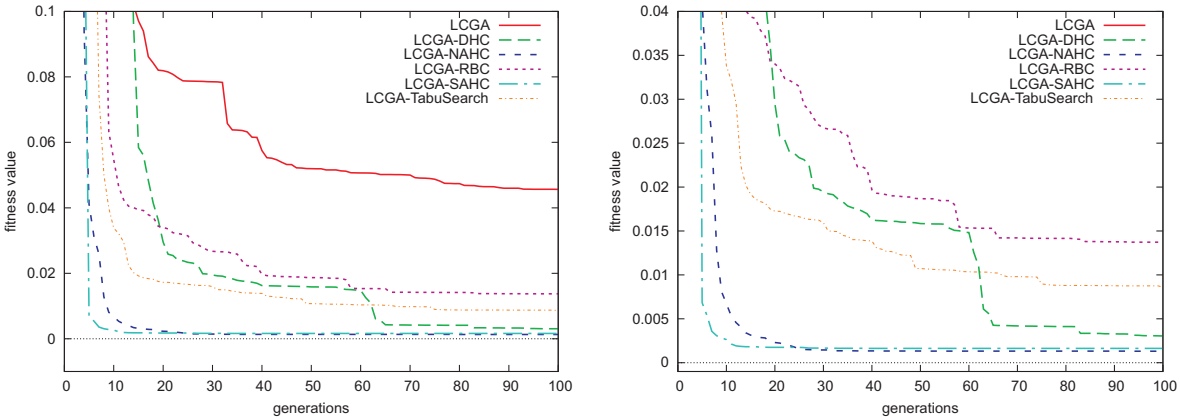


Figure 7.13: Rosenbrock n = 10, LCGA vs hLCGA with population rate exchange strategy and restricted local search

We can graphically observe that using hLCGAs with populations rate exchange strategy combined with restricted local search provides less good results than with complete local search. However all the hLCGA variants still outperform the LCGA but none of them reaches the global optimum. The best result is reached but LCGA-NAHC which is also the fastest one it terms of computational speed, with an average of 6 seconds per experiment (like for the LCGA-RBC which shows the worst result among the hybrid LCGAs). The most computational time demanding is the LCGA-SAHC which shows the fastest convergence speed and the second best solution after the LCGA-NAHC.

The results presented in fig. 7.14 represent the averaged best of generation over 30 experiments for LCGA and hLCGA using the best individual exchange strategy combined with restricted local search (left side) and complete local (right side).

The main difference which appears in both cases compared to the hLCGA with population rate exchange strategy is that one hybrid variant does not perform better than LCGA (LCGA-SAHC when using the restricted local search) or converges slower than LCGA (LCGA-RBC when using complete local search).

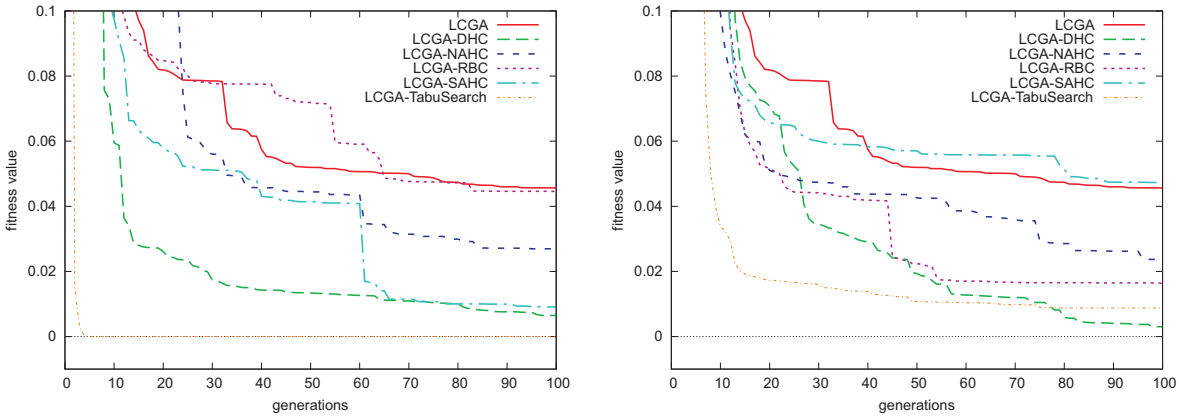


Figure 7.14: Rosenbrock n = 10, LCGA vs hLCGA with best individual exchange strategy and complete (right)/restricted (left) local search

The best result is obtained by the LCGA-TS for the complete local search case and by the LCGA-DHC for the restricted local search case (LCGA-TS shows the best convergence speed in that case).

Concerning the computational speed, using the best individual exchange strategy involves less evaluations for the local search algorithm and thus requires less time. Indeed the difference is minimum with the LCGA except when using the LCGA-TS which is again slower.

As a conclusion, we can deduce that using hLCGA with population rate exchange strategy combined with complete local search provides the best results. For this reason we decided to use this combination on the ICP problem optimization which we present in the following section.

7.5.1.2 ICP problem optimization using hLCGA

Based on the results obtained on the Rosenbrock function optimization problem, we compared the performance of the LCGA to the hLCGA using population rate exchange strategy and complete local search on the ICP optimization problem.

In order to evaluate the influence of the rate of the population which is sent to the local search algorithm, we decided to compare two different rates which are 0.35 (as for the Rosenbrock function) and 0.5.

In order to conduct these experiment the following parameters were used:

Contrary to the ICP instance which has been optimized in 7.4.1, the instance which has been optimized in order to compare the LCGA to the hLCGAs does not include any transportation cost. Therefore the best costs found and presented in table 7.6 and in fig. 7.15 are significantly lower than

		Algorithm	Avg. Time per Exp.	Result
		LCGA	3s	0.0456
Best Individual Improvement	Complete LS	LCGA-SAHC	2s	0.0091
		LCGA-NAHC	2s	0.0269
		LCGA-RBC	2s	0.0446
		LCGA-DHC	3s	0.0065
		LCGA-TS	8s	0.0185
	Restricted LS	LCGA-SAHC	2s	0.0473
		LCGA-NAHC	2s	0.0237
		LCGA-RBC	2s	0.0163
		LCGA-DHC	2s	0.0030
		LCGA-TS	8s	0.025
Population Rate Improvement	Complete LS	LCGA-SAHC	11s	6.3623E-5
		LCGA-NAHC	5s	0.0
		LCGA-RBC	4s	1.2813E-5
		LCGA-DHC	20s	0.0
		LCGA-TS	27s	0.0
	Restricted LS	LCGA-SAHC	9s	0.0016
		LCGA-NAHC	6s	0.0013
		LCGA-RBC	6s	0.0137
		LCGA-DHC	7s	0.0030
		LCGA-TS	7s	0.0086

Table 7.4: Results of all experiments for the Rosenbrock function

in 7.4.1.

As shown in table 7.6 and in fig. 7.15, all hLCGAs outperform the standard LCGA in terms of best result found. The same drawback, as for the Rosenbrock function optimization, concerning the computational time can be noticed. Additionally, due to the higher complexity of the problem, the difference between the standard LCGA and the hLCGAs is even more noticeable (an average of 7 times more time required).

When comparing the results obtained with the two different population exchange rates, it clearly appears that all the results are improved using a higher rate (i.e. the results are better using 0.5 compared to 0.35). As a consequence, since the local search algorithms have more evaluations to execute when using a higher population rate, the time required is also higher (approx. 25% more).

As for the Rosenbrock function optimization using population rate exchange strategy and complete local search, the LCGA-NAHC provides the overall best results with 1148.63$ with a population rate set to 0.35 and 1147.46$ with a population rate set to 0.5. The same conclusion can be drawn for the

Number of Subpopulations	3
SubPopulation size	100 individuals
Termination Condition	100 generations
Representation	16 bits binary
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual
Local search algorithms	SAHC, NAHC, RBC, DHC and TS
Local search termination condition	Complete Search, Restricted Search
Local search exchange strategy	population rate = 0.35 and 0.5

Table 7.5: Parameters used for LCGA and hLCGA on the ICP optimization problem

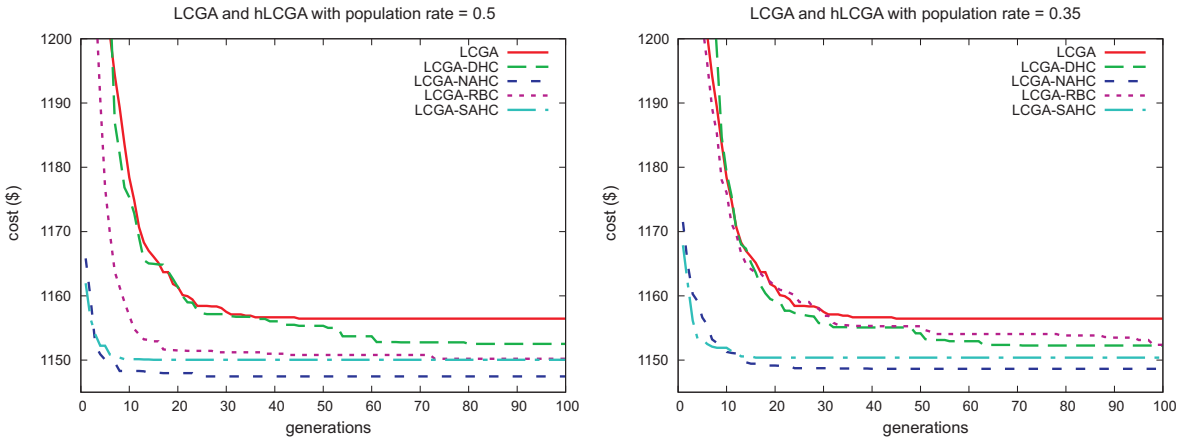


Figure 7.15: LCGA vs. hLCGA on the ICP problem with population rate = 0.5 and 0.35

		Algorithm	Avg. Time per Exp.	Result
		LCGA	25s	1156.46
Population Rate, Complete LS	0.35	LCGA-SAHC	4min26s	1150.39
		LCGA-NAHC	2min42s	1148.63
		LCGA-RBC	2min20s	1152.38
		LCGA-DHC	2min34s	1152.52
	0.5	LCGA-SAHC	3min51s	1150.04
		LCGA-NAHC	3min24s	1147.46
		LCGA-RBC	3min4s	1150.20
		LCGA-DHC	3min13s	1152.27

Table 7.6: Results of all experiments for the Rosenbrock function

least performing algorithms which again are the LCGA-RBC and the LCGA-SAHC. Indeed, not only they provide the worst results among the hLCGAs but their convergence speed is very close to the LCGA one.

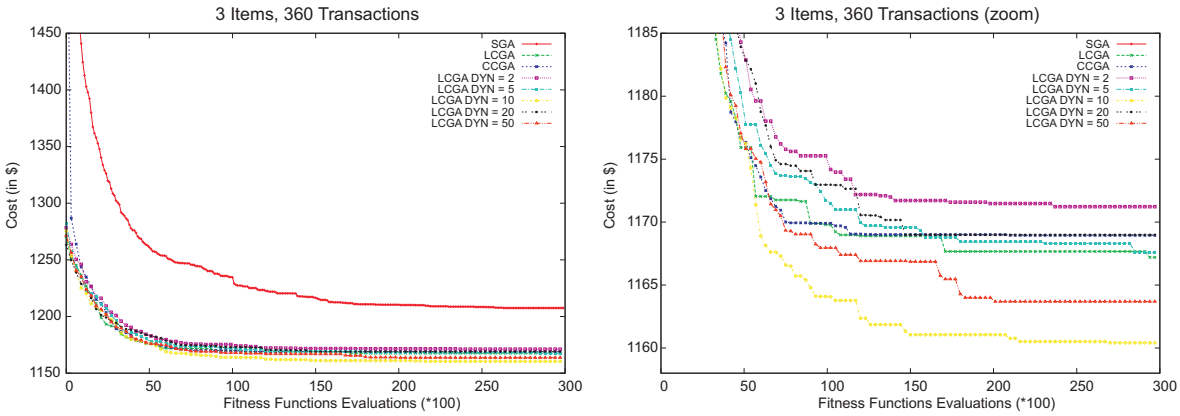


Figure 7.16: ICP optimization with 3 items and 360 transactions using dLCGA

7.6 dLCGA

dLCGA, a new dynamic LCGA, has been described in chapter 5.3. In the following section we provide some experimental results on the ICP optimization problem, in which we compare the performance of dLCGA using several reorganization steps to the "standard" LCGA and CCGA.

7.6.1 Experimental Results

In order to compare the performance of the CCGA, the LCGA and the dLCGA on the ICP problem, the following parameters were set for all the algorithms: population(s) size was equal to 100, $p_k = 0.6$ (crossover probability) and $p_m = 1/\text{chromosome\_length}$ 0.05 (mutation probability).

Number of Subpopulations	3, 100
SubPopulation size	100 individuals
Termination Condition	30000 (3 items), 500000 (100 items) function evaluations
Representation	16 bits binary
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.6
Mutation operator	bit flip, $p_m = 1/\text{chromosome\_length}$
Elitism	1 individual
Reorganization steps	each 2, 5, 10, 20, 50 generations

Table 7.7: Parameters used for LCGA and hLCGA on the ICP optimization problem

Figure 7.16 shows the results averaged on 25 runs with the four algorithms for 360 transactions and 3 types of stock items. dLCGA has been tested using various reorganization steps (e.g. n=5 means that the ring is modified each 5 generations).

It is clear that all CGAs outperform the SGA both in terms of speed of convergence and in the minimum cost found. Compared to CCGA, LCGA converges a bit faster but gets stuck in a local

optimum. Comparing the dLCGAs to the standard LCGA, both results and convergence speed are improved when using n=10 and n=50 but degraded with the other reorganization steps (i.e. 2, 5, and 20).

ICP Parameters	Algorithm	Result
3 Items, 360 Transactions	SGA	1207.47
	CCGA	1168.95
	LCGA	1167.20
	dLCGA, step = 2	1171.23
	dLCGA, step = 5	1167.61
	dLCGA, step = 10	1160.41
	dLCGA, step = 20	1168.98
	dLCGA, step = 50	1163.69
100 Items, 12000 Transactions	CCGA	43509.21
	LCGA	54186.26
	dLCGA, step = 2	53862.77
	dLCGA, step = 5	53921.60
	dLCGA, step = 10	53962.09

Table 7.8: Results of the SGA, CCGA, LCGA and dLCGA on the ICP problem

Figure 7.17 presents the results obtained on a bigger instance of the problem (i.e. 100 items and 12000 transactions). It confirms that the dLCGA performs better than LCGA, for which one explanation can be that exchanging different information after each reorganization permits to improve the global result. One noticeable difference is that the convergence speed and the best result found are improved using all the reorganization steps (2, 5 and 10). Finally, dLCGA is clearly beaten by CCGA. This difference comes from the problem size and the decomposition used for the LCGA and dLCGAs. Indeed, when optimizing the ICP problem for 100 items and 12000 transactions, each population needs to process 240 transactions for the (d)LCGA as opposed to the 12000 transactions processed by each population in the CCGA. On the one hand, the drawback for the (d)LCGA is thus a loss of information on the problem which leads to a local optimum but on the other hand the computational time required is much lower.

Additionally, in [68], it has been demonstrated that LCGA and CCGA have different properties that made them fit for different classes of problems. CCGA seems therefore to be more adapted on big instances of the ICP problem.

7.7 Conclusion

This chapter allowed to illustrate the application of the LCGA and two new variants we have developed, a hybrid one (hLCGA) and a dynamic one (dLCGA), on a stock management problem called Inventory

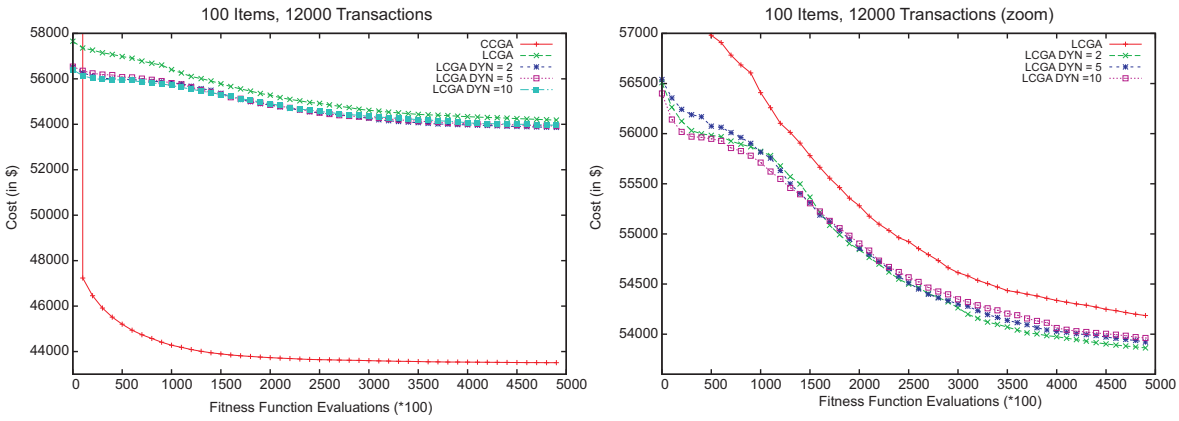


Figure 7.17: ICP optimization with 100 items and 12000 transactions using dLCGA

Control Parameter (ICP) problem. Using LCGA implies decomposing the optimization problem, therefore we chose that, like for the CCGA, one population optimizes the parameters for one item, but contrary to CCGA in which the communication graph is a complete graph, we used a ring topology (as explained in section 7.3. We experimentally showed in section 7.4 that using this decomposition in LCGA provides better results than the CCGA on small instances of the ICP problem (i.e. 2 items and 240 transactions) but its performance degrades as the problem size increases (i.e. with 10/100 items and 1200/12000 transactions). In sections 5.2 and 5.3 we demonstrated that the two new variants of LCGA we have created, respectively hybrid (hLCGA) and dynamic (dLCGA) allow to improve the "standard" LCGA's performances on this business problem.

Chapter 8

Dynamic Problem Case Study: Injection Networks

Contents

Multi-hop ad hoc networks are networks composed of communicating devices capable of spontaneously interconnecting without any pre-existing infrastructure. Among the possibilities of utilization, we can cite: positioning in cities, gaming, tourism, art galleries, etc. The most popular wireless networking technologies available nowadays for building such networks are Bluetooth and IEEE802.11 (Wi-Fi). Devices in range to one another communicate in a point-to-point fashion. Such ad hoc networks are intrinsically dynamic. Due to their limited transmission range, these networks face partitioning problems that penalize their global efficiency. In real scenarios, one or more additional remote links have to be created to keep connected the different clusters of locally interacting users that dynamically move.

In this chapter we consider the problem of optimizing the addition of such long-range links (e.g. GSM, UMTS or HSDPA), are also called *bypass links*, to inter-link network partitions. To tackle this topology control problem, we use small-world properties as indicators for the good set of rules to maximize inter-link efficiency. Small-world networks [158] feature a high clustering coefficient (γ) while still retaining a small characteristic path length (L). A small path length corresponds to fewer hops, which is of importance for effective routing mechanisms as well as for the overall communication performance of the entire network. The clustering coefficient represents the connectivity in the neighborhood of each node and thus reflects the degree of information dissemination each single node can achieve. This finally motivates the objective of evoking small-world properties in such settings.

In order to optimize those parameters (maximizing γ, minimizing L) and to minimize the number of required bypass links in the network, we here rely on Evolutionary Algorithms (EAs) [159] and more specifically on Genetic Algorithms (GAs) [33].

The remainder of this chapter is organized as follows. In the next section we introduce the latest key related works. In Section 8.2 we give a detailed view on the injection network problem. Section 8.4 provides details on our three algorithms, considering two representations and two crossover operators. The section is concluded by a detailed description of the fitness function we defined. Then, in Section V, we present the experiments and discuss the results. The last section contains our conclusions and perspectives.

8.1 Related Works

Mobile multi-hop ad hoc networks have brought several difficult (and practical) challenges. Among them, we will focus on network partitioning which is of importance for communication performance as well as routing mechanisms. Some past works advise the utilization of hybrid wireless networks, where a fixed infrastructure supports a higher connectivity among several clusters of ad hoc networks and avoids network partitioning [160] [161] [162]. However, such hybrid wireless networks are often

not feasible, because of economical and implementation issues.

In order to tackle the same problem, some recent researches started investigating the advantages of bringing small-world properties to such networks. Let us highlight the following ones. Dousse in [163] introduces base stations, connected through a fixed wired infrastructures, in order to increase connectivity in ad hoc networks, thus realizing global reachability. Helmy [164] considers using the uniform distribution (of random links) with which the objective is to reduce the number of queries during the search for a given target node. In a similar way, Reznik et al. study the effet of randomly adding point-to-point wired links in a square grid ad hoc network in [165]. They provide an elaborate framework with a specific parameterized distribution for choosing long links. This helps to reduce the path length to a small power of the initial diameter. In his PhD dissertation [166], Nguyen investigates the use of small-world graphs for network design. To this end, he extends Reznik's framework by adding a cost (weight of long links) and congestion but he still studies square grid networks and his long-links construction scheme implies a high number of them. In [167], Sharma also investigates the use of small-world properties for the addition of wired links in hybrid wireless sensor networks. His objective is to reduce the energy dissipation per node.

Alternatively, an infrastructureless setting is of interest where problems of restricted geographical regions are avoided. Watts [158] introduces a spatially defined link, called *global edge*, with length-scaling properties to include spatial models in his investigations. Some approaches extend standard ad hoc network models, by considering two different transmission ranges [168] [169], e.g. small distance Bluetooth links along with higher distance Wi-Fi links.

Some other researches focus on the optimization of this topology control problem using evolutionary algorithms, including genetic algorithms. In [170], Lee uses a multi-objective genetic algorithm, named GA-OTC, for topology control in wireless sensor networks. The algorithm searches for the optimal clustering set of sensor nodes so as to fulfil the objectives which are to balance and minimize energy consumption. In [171], Pandey uses a genetic algorithm to optimize the placement of special multi-interface devices called *drones* in an ad hoc heterogeneous network comprising of several underlying homogeneous networks. p drones have to be placed to improve the network connectivity while minimizing the network partitions and the number of interfaces on each drone.

However, in the current literature there is no research merging both the optimization of such hybrid ad hoc networks with the small-world properties of such networks. This has motivated the definition of the problem we introduce in the following section.

8.2 Injection Networks Problem Description

The problem we study in this chapter, consists in overcoming partitioning in ad hoc networks by optimizing the placement of long range links that we call *bypass links*.

Our initial motivation for the current investigation is based on the assumption that technologies like Bluetooth and Wi-Fi can be used to create ad hoc communication links within the transmission range at no charge. Additional cellular network links such as GSM/UMTS/HSDPA might be employed by appropriately equipped devices to establish supplementary communication links, that we call *bypass links*, between two arbitrary devices. These links will induce additional costs. Let us formalize the notion of bypass links.

Definition 1 (Watts) *The spatial neighborhood $\Gamma_{tr}(v)$ of a node v is the set of nodes within transmission range tr of v.*

Definition 2 *A bypass link is a link (u,v) between nodes u and v with $u \notin \Gamma_{tr}(v)$.*

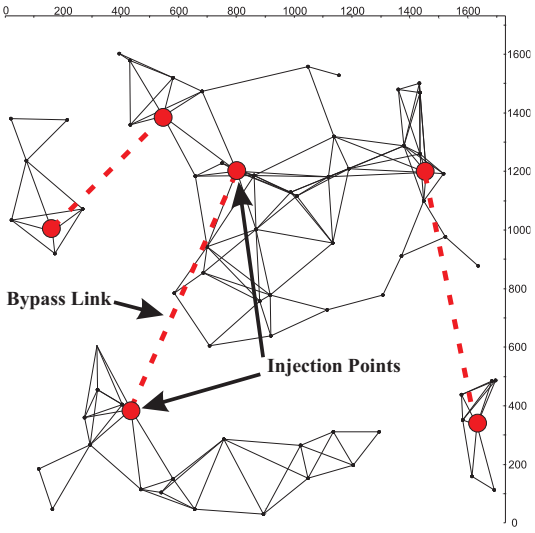


Figure 8.1: Example of an Injection Network

That is, a bypass link is a link which connects two nodes that are not in the same spatial neighborhood. Please note that elements of $\Gamma_{tr}(v)$ do not necessarily have to be connected to v in real settings. Practically, a bypass link can be built by using a cellular network as well as by using access points. Nevertheless, in our model a bypass link is counted as a single hop, thus simplifying the real topology behind that bypass link.

The injection communication paradigm is based on establishing bypass links between carefully selected devices. Herrmann et al. [172] called these dedicated communication points *hub nodes*. We call these dedicated devices used for establishing bypass links *injection points*.

Definition 3 *Two nodes u and v are called injection points if a bypass link (u,v) exists between nodes u and v.*

Injection points serve two different purposes: a point where information dissemination starts and where services are being placed (service placement, Herrmann et al. [172]). In the first case, the injection point is of essential importance at the moment of receiving information and passing this information to the neighborhood. The injection point might represent a bottleneck, depending on the amount of data passing through. In addition, injection points become particularly attractive when offering a service. In fact, information dissemination can be seen as such a service that is usable by devices in the injection points surroundings. Different criteria for determining the injection point can be of importance. Supposing that, for instance, the device is highly clustered and thus one of the central members of a group, epidemic behavior for information spreading will take effect faster. Therefore, the current environment and the device's relationship to its neighbors are important.

For self-organizing communication networks based on bypass links and injections points as described before we use the term *injection networks*.

8.2.1 Small-Worlds

In this optimization problem, we consider small-world properties as indicators for the good set of rules to maximize the bypass links efficiency. Small-World networks [158] are a class of random graphs that exhibit a small characteristic path length (L), indicating the degree of separation between the nodes in the graph, and a high clustering coefficient (γ), defining the extent to which nodes in the graph tend to form closely-knit groups that have many edges connecting each other in the group, but very few edges leading out of the group. The challenging aspect in using small-world properties is that small-world networks combine the advantages of regular networks (high clustering coefficient) with the advantages of random networks (low characteristic path length).

Small-world networks [158] are a class of random graphs that exhibit two main characteristics: a small characteristic path length (L) and a high clustering coefficient (γ). A formal definition of these two graph measures is given below:

Definition 4 (Watts) *The local clustering coefficient γ of one node v with k_v neighbors is*

$$\gamma_v = \frac{|E(\Gamma_v^r)|}{\binom{k_v}{2}}$$

where $|E(\Gamma_v^r)|$ *is the number of links in the relational neighborhood of* v *and* $\binom{k_v}{2}$ *is the number of possible links. The clustering coefficient is the average local clustering coefficient for all nodes of a network.*

For example, in Figure 8.2, node a is connected to three nodes b, d and e. The maximum number of possible edges among these three nodes is three. The graph shows that only two out of those three possible edges exist (between b-e and d-e). The edge b-d is missing. So the clustering coefficient for node a is 2/3 or about 0.67. For Figure 8.2, this value is 0.67. In a physical sense, the clustering coefficient defines the extent to which nodes in the graph tend to form closely-knit groups that have many edges connecting each other in the group, but very few edges leading out of the group.

Definition 5 (Watts) *The shortest path length* $\overline{d}_v$ *connecting each node* $v \in V(N)$ *of a network* N *to all other nodes is* $d(v,j)\ \forall\ j \in V(N)$*. The characteristic path length* L *is the median of all shortest paths.*

The characteristic path length is a measure of the number of hops necessary to reach any node in the network from any other node. This indicates the degree of separation or connectivity between nodes in the graph. In Figure 8.2, node a can reach three of the nodes (b, d and e) through just one hop and the fourth node (c) via two hops. So the characteristic path length for this node is:

$$(a) = \frac{HopsToReachAllNodes}{NumberOfNodes} = \frac{(3\times1)+(1\times2)}{4} = 1.25$$

The characteristic path length (L) for the entire graph, which is the mean of the characteristic path length of all nodes, is equal to 1.2. The challenging aspect in using small-world properties is that

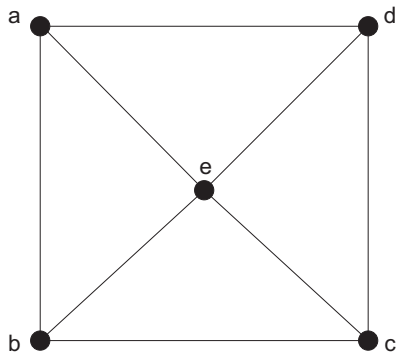


Figure 8.2: Graph with $\gamma = 0.67$ and L = 1.2

small-world networks combine the advantages of regular networks (high clustering coefficient) with

the advantages of random networks (low characteristic path length).

In order to study the small-world properties of such hybrid networks, we had to rely on some ad hoc network simulator. In our case we used Madhoc [32], an application-level network simulator dedicated to the simulation of mobile ad hoc networks. The main motivation for using Madhoc is its ability to simulate hybrid networks, i.e., mixing different technologies (e.g., bluetooth/Wi-Fi for local connections and UMTS for long distance calls), and its graphical and batch modes of visualization, which greatly help in understanding the network design alternatives.

8.3 Fitness Function

As stated before, we relied on small-world networks properties so as to optimize the placement of the bypass links. We have conducted our experiments using the Madhoc simulator which to simulate and to visualize hybrid ad hoc networks (using Wi-Fi, bluetooth, GSM, UMTS). We extended Madhoc in order to make it support bypass links and to measure small world properties.

In order to assign a fitness to the candidate solutions (i.e. sets of possible bypass links) of our algorithms, we use a unique cost function F which combines the two small world measures (L and γ) and the number of created bypass links.

When computing the fitness function, we first test if the global network is connected. Indeed, since we use small-world properties as indicators, the network has to be connected in order to compute the characteristic path length (L) on the global network. If the optimized network is not connected, due to too few or not efficiently placed bypass links, the fitness value is a weighted term of the number of partitions in the network. On the contrary, if the network is connected, the fitness value is a linear combination of the small world measures (clustering coefficient and characteristic path length) and of the difference between the number of bypass links and the maximum number allowed. We look for maximizing the clustering coefficient and minimizing both characteristic path length and number of bypass links. Using this fitness function we consequently have a maximization problem as defined in Algorithm 7.

Algorithme 7 : Fitness Function

if *Graph connected* **then**
  F = α * γ - β * (L - 1) - δ * (bl - bl_{max})
else
  fitness = ξ * P
end

With weights experimentally defined:
$\alpha = 1$
$\beta = 1$ /(N -1)
$\delta = 2$ / (N * (N-1))
$\xi = 0.1$

bl is the number of bypass links created in the simulated network by one solution, bl_{max} (defined a priori) is the maximum number of bypass links that can be created in the network, P is the number of remaining partitions in the whole network after the addition of bypass links and N is the number of stations in the global network.

8.4 Static Injection Network Optimization

8.4.1 CCGA vs. Generational and Steady State GAs

In this section we first introduce the three genetic algorithms we used. Next we provide details on the two solution encodings and on the two crossover operators we applied. Finally we present the fitness function we have defined.

The use of evolutionary computation (EC) techniques to evolve solutions for both abstractions and real-life problems has seen a dramatic increase in popularity and success over the last decade. The most popular and widely applied EC technique is the sequential GA [33], whose computational scheme is based on a set (population) of potential solutions (individuals) on which it applies some stochastic operators in order to search for an optimum. It uses a single population (panmixia) of individuals and apply operators to them as a whole.

Past works have shown that the underlying iterative step of the GA is very influent in some applications [34]. Therefore in this work we focus on a generational, a steady-state and a cooperative coevolutionary GA.

Since this problem is new to the metaheuristic community, we start by investigating the kind of evolution step more amenable to our problem by analyzing three proposals: a generational GA [52], a steady-state GA [6], and a cooperative coevolutionary GA [11] on three different instances of a partitioned ad hoc network. We further will investigate the influence of the solution representations and of the crossover operators on the final quality of the results, as an important methodological step in applying GAs to complex problems.

8.4.1.1 Solution Encodings

Solution encoding is a major issue in this kind of algorithms since it will determine the choice of the genetic operators applied for exploring the search space.

First Encoding

We have used a binary encoding of the solution in which each gene encodes an integer on 15 bits, that corresponds to one possible bypass link in the half-matrix of all possible links. For instance, if the maximum number of bypass links fixed a priori for the network that is optimized is 10, then a chromosome will have 10 genes of 15 bits. Figure 8.3 shows the example of a chromosome composed of 2 genes (thus the maximum number of created bypass links is 2) on a network of 5 stations. The 5×5 half-matrix represents all the possible links in the network including the already existing local links in the network (i.e. the existing Wi-Fi connections). In the example showed in Figure 8.3, the first gene (circled) with the integer value 2 stands for the connection between station 1 and station 3 in the corresponding half-matrix (also circled).

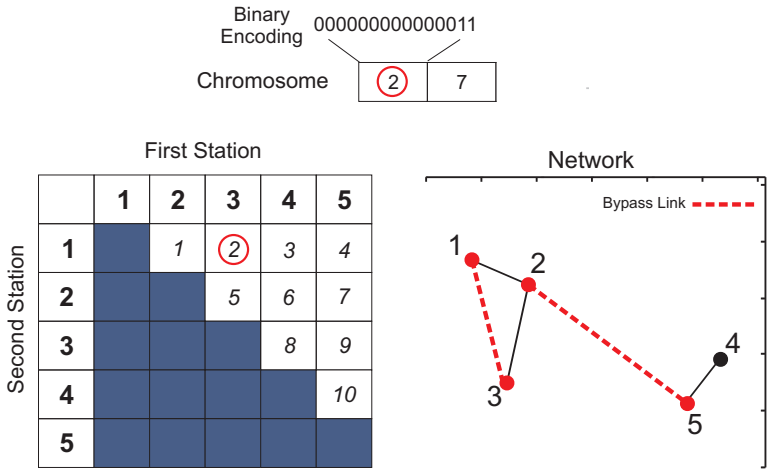


Figure 8.3: Example of the First Solution Encoding

Second Encoding

The second encoding is also binary. Each bit represents one possible bypass link in the network. If the bit value is 1 then the corresponding bypass link is created and if it is 0 it is not created. Let us take as example the same network as in Figure 8.3, in which the network is composed of 5 stations and 3 existing wireless links. The number of possible links in this network is

$$\frac{(N) * (N-1)}{2} = \frac{5 * 4}{2} = 10 \tag{8.1}$$

with $N =$ numberOfNodes. The number of possible bypass links is finally 10 - 3 = 7. The resulting number of bits in the chromosome is 7, as shown in Figure 8.4. The half-matrix represents the possible

links in the network and the light gray shaded cells represent the already existing wireless links that are not considered (i.e. links between stations 1-2, 2-3 and 4-5). In our example, the first bit in the chromosome thus stands for the possible bypass link between station 1 and station 3, which is created since its value is 1, the second bit stands for the link between station 1 and station 4, this one is not created since its bit value is 0, and so on. Contrary to the first encoding, this second encoding depends on the network size and on the number of existing links. Consequently, the bigger the network, the bigger the chromosome.

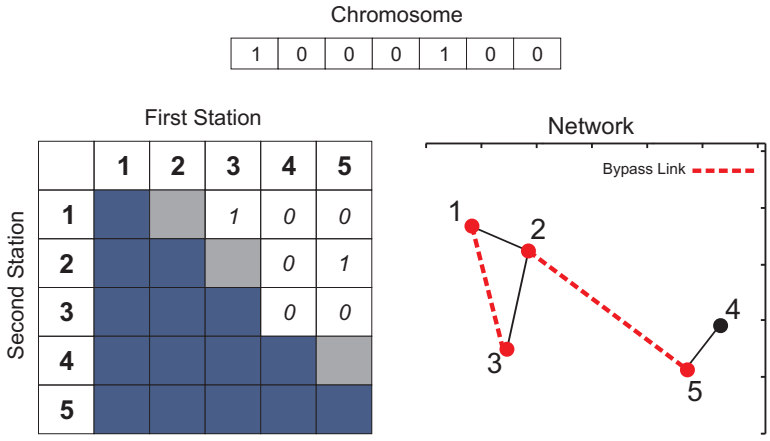


Figure 8.4: Example of the Second Solution Encoding

8.4.1.2 Crossover Operators

The crossover operation, also named as recombination, produces new individuals by combining the information contained in two or more parents. This is done by combining the values of the two parents. Our experimentations have been conducted using both two-point crossover and uniform crossover to further investigate the interest and the influence of each operators using our two representations for the injection network problem. A description of these two genetic operators can be found in 2.1.1.4.

8.4.1.3 Experimentation

This section presents the results obtained on the injection network optimization problem using the three GAs, generational, steady state and cooperative coevolutionary. We first describe the parameters used for the genetic algorithms. Next, the configuration of the network simulator is introduced and, finally the results obtained using the three GAs, using the two representations and the two crossover operators (2-point and uniform), are analyzed and compared.

The algorithms have been implemented in Java and tested on a 3.2 GHz Xeon processor with 4 GB of RAM, running Debian Linux (with kernel 2.6.9-22) and Java version 1.5.0_05.

GA Parameterization

Population size	100 indiv. (10x10 for CCGA)
Termination Condition	50,000 function evaluations
Selection	Binary Tournament
Crossover operators	2-point and uniform, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual (not for ssGA)

Table 8.1: Parameters for genGA, ssGA and CCGA

In table 8.4, we show the parameters used for genGA, ssGA and CCGA.

We used a randomly generated population composed of 100 individuals for genGA and ssGA and 10 subpopulations of 10 individuals for CCGA. The selection operator for genGA and CCGA is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). For ssGA we have used a replace-worst strategy. As stated before, the two crossover operators (separately) analyzed are 2-point and uniform crossover used with probability p_c=0.8. The mutation operator is bit flip mutation in which each allele of the chromosome is flipped with probability p_m= 1/chromosome_length. Concerning the generational GA and CCGA we have added elitism: the best individual found in one generation is thus kept for the next generation.

Madhoc Configuration

As stated before, the Madhoc simulator was used for managing the complex scenarios posed by this injection network problem. Figure 8.5 shows how the genetic algorithms interact with Madhoc.

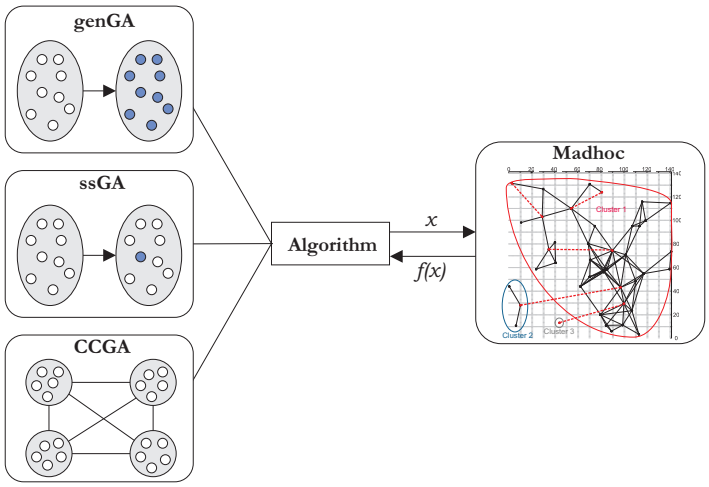


Figure 8.5: Components of the experimental study

We have defined a squared simulation area of 0.2 km^2 and tested three different densities of 150, 210 and 350 devices per squared kilometer. Each device is equipped with both Wi-Fi (802.11b) and

UMTS technologies. The coverage radius of all mobile devices ranges between 20 and 40 meters in case of Wi-Fi.

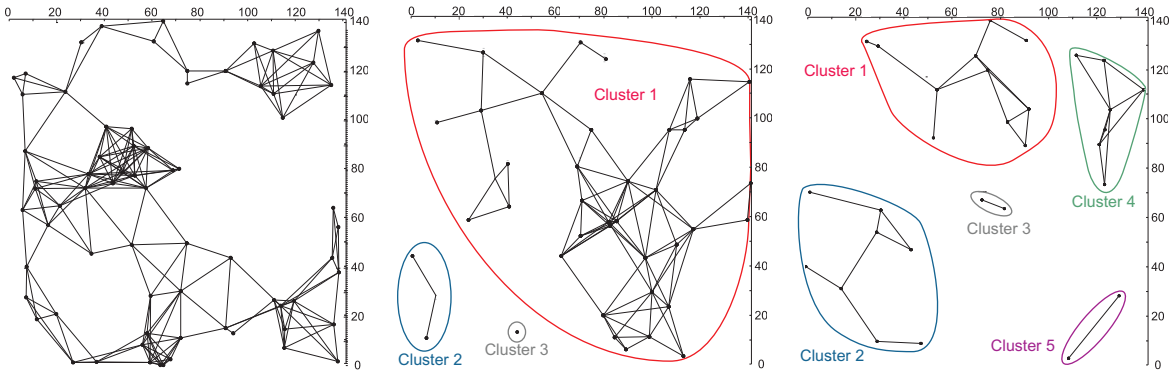


Figure 8.6: Studied Networks with 1, 3 and 5 clusters

The studied networks, as presented in Figure 8.13, here represent a snapshot of mobile networks in the moment in which a single set of users moved away from each other creating the clusters of terminals, that were obtained using the graphical mode of Madhoc. As an example, the network with 3 clusters (center of Fig. 8.13) consists in 42 stations located in three partitions, the first partition has 38 stations, the second one 3, and the third one has a single station. The number of possible connections in this 3-clusters network is $\frac{N*(N-1)}{2} = 861$. The number of existing Wi-Fi connections in this network is 116, thus the number of possible bypass links is 861-116 = 745. The clusters are selected purposely to be different and thus challenging.

	1 Cluster	3 Clusters	5 Clusters
Surface	0.2 km^2	0.2 km^2	0.2 km^2
Node Density	$350/km^2$	$210/km^2$	$150/km^2$
Number of Nodes	70	42	30
Partitions	1	3	5
Possible Links	2189	745	400

Table 8.2: Parameterization used in Madhoc

Results

Each result presented hereafter is the average obtained on 30 independent runs. In order to establish the statistical significance of the means, we first have checked that the data is normally distributed using the Kolmogorov-Smirnov test. If so, we then perform an ANOVA test so as to compare the means otherwise we use a Kruskal-Wallis test [41].

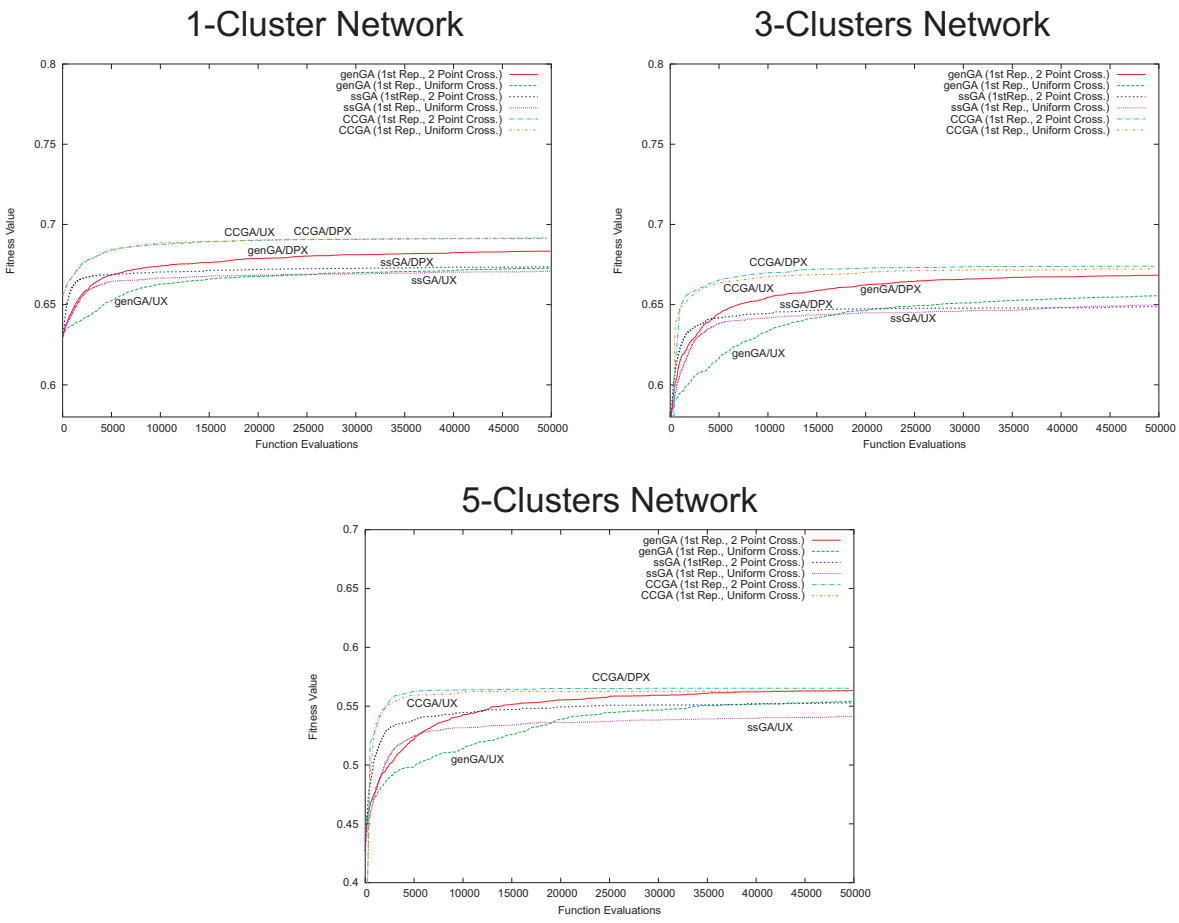


Figure 8.7: Average results of 30 runs on the 3 networks using the first representation

In Table 8.3 we show the averaged results and the total computational time for all 30 runs for each algorithm, representation and crossover operator.

Comparison of the Algorithms

Comparing the three algorithms in terms of best results found, it clearly appears that CCGA outperforms the two panmictic algorithms by reaching the highest fitness for the three network scenarios (see Table 8.3). We can also see that ssGA always performs better than genGA. Another interesting property is that the difference between the genGA and the ssGA increases as the number of simulated stations decreases (see 8.7 and 8.8). Between ssGA and CCGA, the behavior is opposite, i.e. the difference decreases as the number of simulated stations decreases. Regarding the execution time required, we see that CCGA requires from two to four times more computational time than the genGA and the ssGA, the bigger difference being observed on the 5-clusters network. The highest computational time is reached on the biggest network (i.e. 1-cluster network) by the CCGA using the second representation and 2-point crossover.

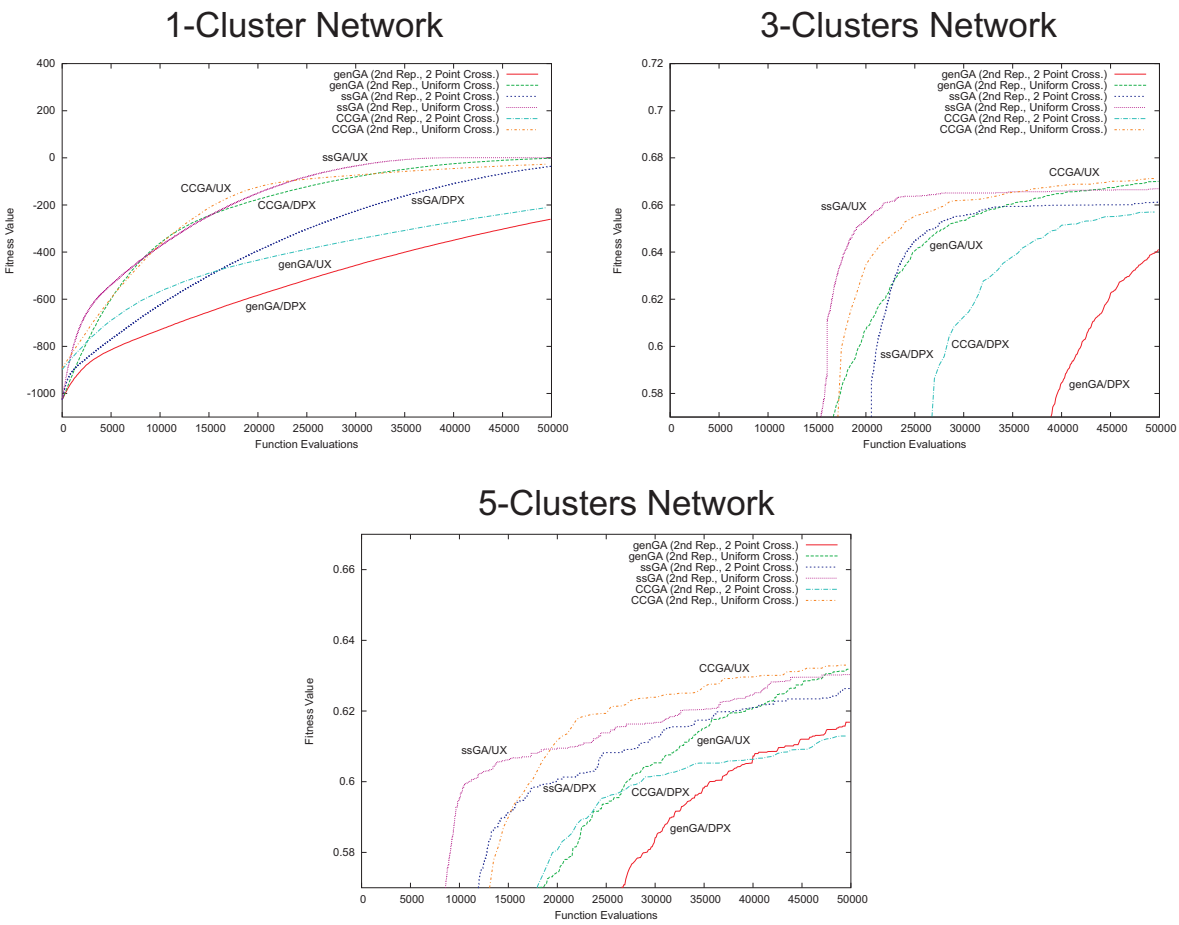


Figure 8.8: Average results of 30 runs on the 3 networks using the second representation

Comparison of the Representation

As can be seen in Table 8.3, using the first representation allows all the algorithms to find a solution in each experiment. Using the second representation highly deteriorates the results when the number of stations in the simulated network is high. Indeed only the ssGA algorithm with uniform crossover manages to find a solution that does not overpass the constraint of 10 bypass links on the 1-cluster network (see top-left corner of Figure 8.8. This difference does not exist when the number of stations decreases, since both genGA and ssGA obtain better results using the second representation on the 3-clusters network (except for the genGA with 2-Point crossover). On the 5-clusters network, the results a slightly deteriorated except for the CCGA with 2-point crossover that obtains the overall best result. One drawback of the second representation is a slower convergence speed than with the first representation. Another disadvantage of the second representation concerns the execution time that is in average twice longer than with the first representation. This is due to the high number of bypass links that are created using this representation.

Network	GA	Repres.	Cross.	Time (s)	Result
1 Cluster	genGA	1st Rep.	DPX	8256	0.6833
			UX	8052	0.6727
		2nd Rep.	DPX	27468	–
			UX	24371	–
	ssGA	1st Rep.	DPX	8395	0.6736
			UX	8950	0.6708
		2nd Rep.	DPX	19151	–
			UX	15273	0.6686
	CCGA	1st Rep.	DPX	17784	0.6911
			UX	17741	**0.6917**
		2nd Rep.	DPX	42579	–
			UX	27280	–
3 Clusters	genGA	1st Rep.	DPX	4486	0.6685
			UX	3483	0.6555
		2nd Rep.	DPX	8937	0.6412
			UX	7354	0.6700
	ssGA	1st Rep.	DPX	3289	0.6489
			UX	4282	0.6500
		2nd Rep.	DPX	4192	0.6612
			UX	5204	0.6669
	CCGA	1st Rep.	DPX	6793	**0.6739**
			UX	10102	0.6723
		2nd Rep.	DPX	16635	0.6570
			UX	13927	0.6685
5 Clusters	genGA	1st Rep.	DPX	1672	0.5634
			UX	1654	0.5539
		2nd Rep.	DPX	3144	0.6168
			UX	2689	0.6318
	ssGA	1st Rep.	DPX	1717	0.5527
			UX	1756	0.5412
		2nd Rep.	DPX	2027	0.6263
			UX	2045	0.6303
	CCGA	1st Rep.	DPX	8674	0.5652
			UX	7206	0.5628
		2nd Rep.	DPX	8799	**0.6329**
			UX	8531	0.5628

Table 8.3: Results of all experiments

Comparison of the Crossover Operators

As it can be seen in Table 8.3, using the uniform crossover provides worse results than 2-point crossover when using the first representation. The contrary is true only when using the first representation for the CCGA on the 1-cluster instance and for the ssGA on the 3-clusters instance. The situation is opposite when using the second representation, since results are better than with the 2-point crossover. The single exception is for the CCGA on the 5-clusters instance which obtain the overall best result applying a 2-point crossover on the second representation. Concerning the influence of the crossover operators on the execution time, we can see that ssGA behaves the opposite way from genGA and CCGA. Indeed ssGA requires more time when using uniform crossover (except on the 1-cluster network with the second representation) contrary to genGA and CCGA (except on the 3-clusters network with the first representation) that are faster with a uniform crossover.

Comparison of the Computational Speed

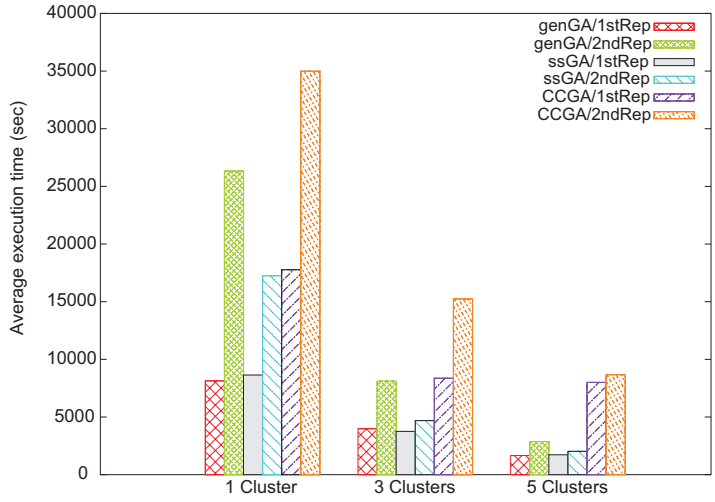


Figure 8.9: Computational speed per algorithm and representation for each network

Figure 8.9 shows the computational speed of each combination of algorithm and representation for the three network scenarios. Each column is thus the average of the computational time obtained using both two-point and uniform crossover. We can clearly observe that the bigger the network the bigger the calculation time. As previously mentioned, we can graphically see that using the second representation always increases the amount of time required. CCGA is always slower than genGA and ssGA. The two panmicitic algorithms perform best using the first representation, the genGA being the overall best algorithm for the 1-cluster and 5-clusters networks and the ssGA for the 3-clusters network. Finally we can notice that in terms of computational time the ssGA is less sensible to the representation than the other two algorithms.

Analysis of the Problem

In this subsection we analyze the most complex problem among the three we have used, i.e. the 1-cluster network. We have studied the evolution of the number of bypass links created by each algorithm according to the number of function evaluations (see Figure 8.10). It is interesting to notice the difference of behavior between the two panmictic GAs and the coevolutionary GA. Indeed both generational and steady state GAs first start by decreasing the number of bypass links (the first 2500 evaluations for the ssGA and between 7500 and 12500 for the genGA) before raising as opposed to the CCGA that directly increases this value. Another noticeable difference is the perturbation that feature the two genGA curves contrary to the ssGA and the CCGA ones. The ssGA with two-point crossover reaches a lower number of bypass links than the genGA with two-point crossover, and additionally provides a better final result. We can also observe that only CCGA reaches the limit of 10 bypass links, which could be considered as a worse behavior than the other two algorithms, but it allows much better final results.

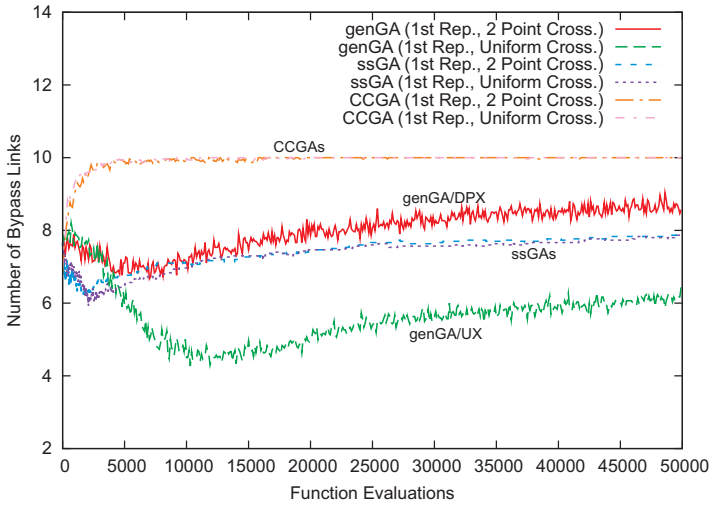


Figure 8.10: Number of bypass links considered in best solution per generation

Our second investigation focused on the stations in the network that are the most often elected as injection points. Figure 8.11 shows the number of times each station has been chosen as injection point in one experiment. This value is the average of all the algorithms that managed to obtain a non-partitioned network (see Table 8.3). The 20 stations that were chosen as best injection point candidate are represented using filled rectangles.

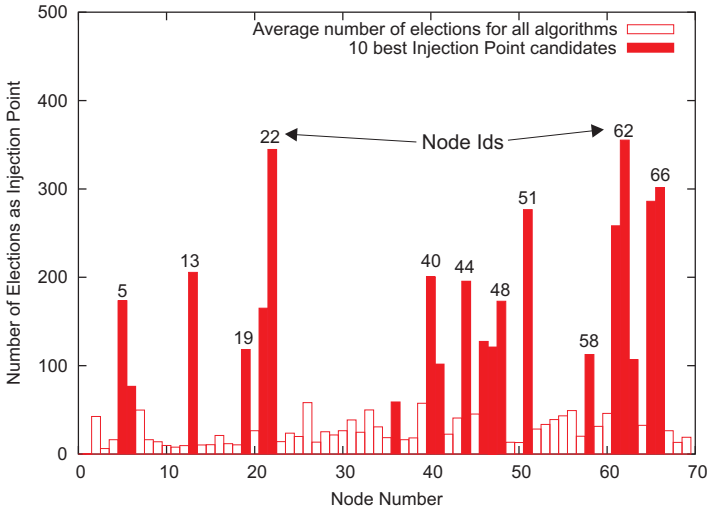


Figure 8.11: Average number of elections as injection point for the 1-Cluster network

Figure 8.12 permits to locate in the network the same 20 stations which were the most often elected as injection point. It is interesting to notice that not only stations that are located inside high density areas are good injection point candidates (e.g. station 41 or 47), but also stations that interconnect high density areas, like station 46 and 61.

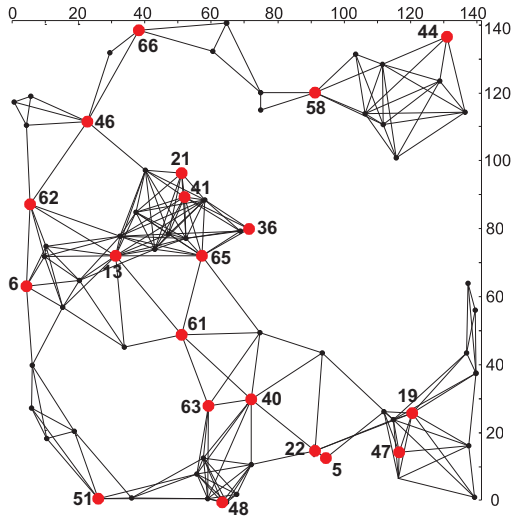


Figure 8.12: Best injection point candidates

8.4.2 LCGA vs. CCGA

This section presents the results obtained on the injection network optimization problem using the the two CGAs (LCGA and CCGA) with different numbers of subpopulations (2, 5 and 10 subpopulations). We also include the results obtained with a generational GA (genGA) and a steady state GA (ssGA) that we use as basis of comparison. We first describe the parameters used for the three genetic algorithms. Next, the configuration of the network simulator is introduced and, finally the results obtained using the CCGA, genGA and ssGA are analyzed and compared.

The algorithms have been implemented in Java and tested on a server with a 3.7 GHz Xeon processor, 16 GB of RAM, running Debian Linux (with kernel 2.6.9-22) and Java version 1.6.0.

8.4.2.1 GA Parameterization

In table 8.4, we show the parameters used for LCGA, CCGA, genGA and ssGA.

Both LCGA and CCGA were tested with 2, 5 and 10 subpopulations. For all algorithms we used a randomly generated population composed of 100 individuals for genGA and ssGA and of 50 individuals for LCGA and CCGA. A solution is encoded as a binary string chromosome, in which each gene encodes an integer on 15 bits (see section 8.4.1.1 for a detailed description). Each gene corresponds to one possible bypass link in the half-matrix of all possible links. The selection operator is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). The crossover operator is uniform. It is used with a probability p_c=0.8. The mutation

operator is bit flip mutation in which each allele of the chromosome is flipped with probability $p_m = 1/chromosome\_length$. Elitism has been added for genGA, LCGA and CCGA: the best individual found in one generation is thus kept for the next generation.

Number of Subpopulations	2, 5, 10 (only for LCGA and CCGA)
(Sub)Population size	100 (genGA, ssGA), 50 (LCGA, CCGA)
Termination Condition	50,000 function evaluations
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.8
Mutation operator	bit flip, $p_m = 1/$chrom_length
Elitism	1 individual (not for ssGA)

Table 8.4: Parameters used for genGA, ssGA, CCGA and LCGA

8.4.2.2 Madhoc Configuration

In order to compare the performance of the CCGA and the LCGA, we used the same 3-clusters network as presented in section 8.4.1.3. This network, as presented in Figure 8.13, represents a snapshot of a mobile network in a squared simulation area of 0.2 km^2 with a density of 210 devices per square kilometer. Each node is equipped with both IEEE802.11b and UMTS technologies. The coverage radius of all mobile devices is set to a random value between 20 and 40 meters.

Surface	0.2 km^2
Node Density	210 / km^2
Number of Nodes	42
Partitions	3
Possible Links	745

Table 8.5: Parameterization used in Madhoc

This network is consists in 42 stations located in three partitions: the first partition consists of 38 nodes, the second one 3, and the third one is made of one single node. The number of possible connections in this 3-clusters network is $\frac{N*(N-1)}{2} = 861$. The number of existing IEEE802.11b connections in this network is 116, thus the number of possible bypass links is 861-116 = 745. This clusters are selected because of their difference in terms of the number of node they host.

8.4.2.3 Results

The results presented hereafter are average out of 30 independent runs. In order to establish the statistical significance of the means, we first have checked that the data is normally distributed using the Kolmogorov-Smirnov test. If so, we then perform an ANOVA test so as to compare the means otherwise we use a Kruskal-Wallis test [41].

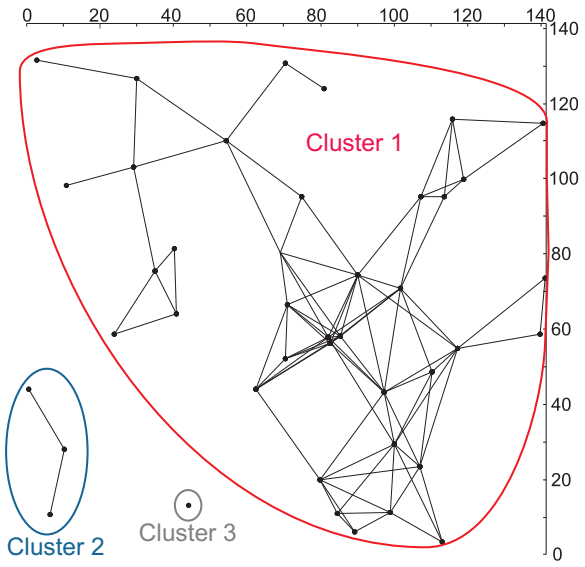


Figure 8.13: Studied Network with 3 clusters

In Table 8.6 we show the averaged results for all 30 runs for each algorithm.

(Sub-)Population	GA	Crossover	Time (s)	Result
1 Pop.	genGA	Uniform	119	0.6555
	ssGA	Uniform	138	0.6500
2 Pop.	LCGA	Uniform	175.62	0.6598
	CCGA	Uniform	217.37	0.6597
5 Pop.	LCGA	Uniform	212.67	0.6634
	CCGA	Uniform	273.54	0.6706
10 Pop.	LCGA	Uniform	268.6	0.6599
	CCGA	Uniform	361.77	**0.6723**

Table 8.6: Results of all experiments

As it can be seen in Table 8.6, using coevolutionary genetic algorithms always provides better results than both genGA and ssGA, ssGA being the least performing one (with statistical confidence). This can be graphically observed in Figure 8.14, as well as the better convergence speed of the cooperative coevolutionary genetic algorithm compared to the other GAs.

When comparing CCGA and LCGA, it appears that CCGA provides better results than LCGA, both in terms of best solution found and convergence speed. This difference increases as the number of subpopulations increases too (see Figure 8.14), the overall best result being obtained by the CCGA with 10 subpopulations.

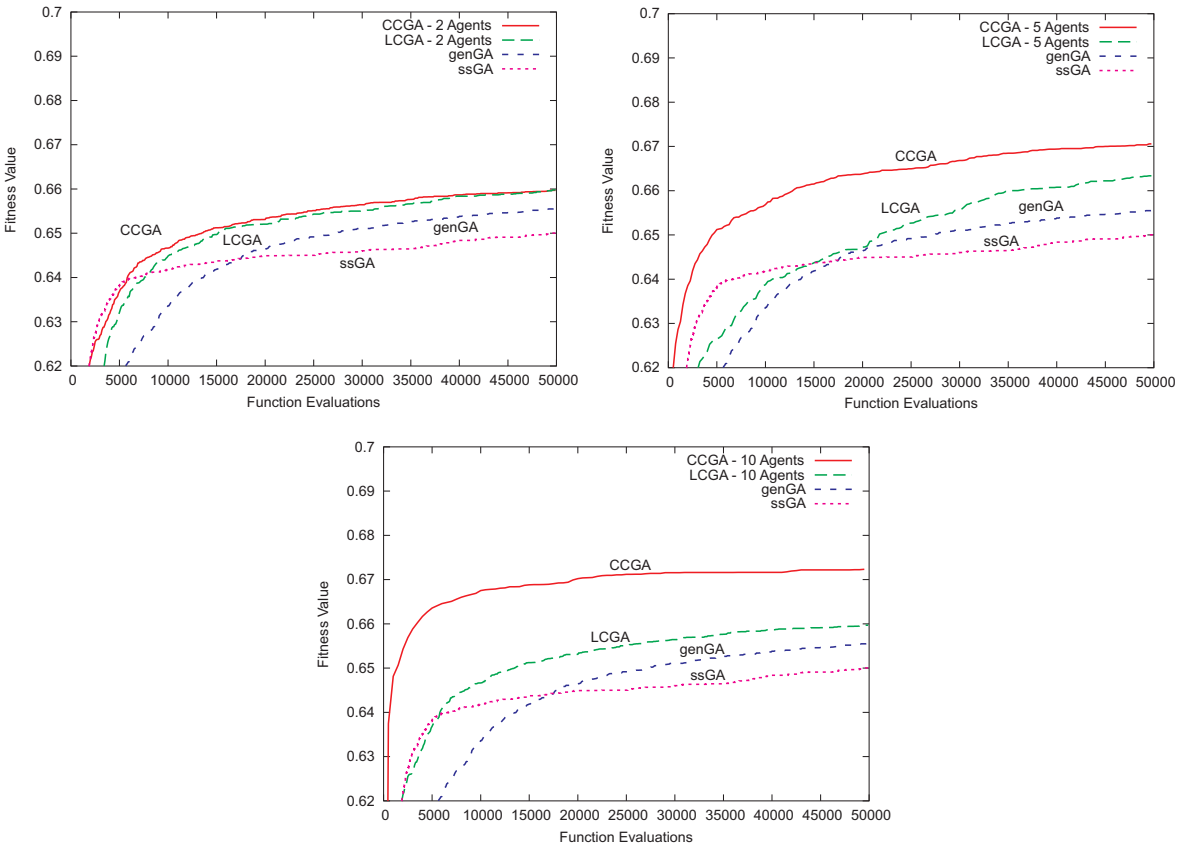


Figure 8.14: Average results of 30 runs using CCGA, LCGA, genGA and ssGA

Regarding the average execution time required per run of the algorithms (see Figure 8.15), we clearly see that CCGA takes more time than LCGA and this difference increases with the number of subpopulations. This is due to the fact that CCGA requires more synchronization than LCGA, indeed in CCGA only one subpopulation is active at one time, which is not the case with LCGA.

8.4.3 Distributed CCGA

This section presents the results obtained on the injection network optimization problem using the distributed CCGA compared to the results given by the generational GA (genGA) and the steady state GA (ssGA). We first describe the parameters used for the three genetic algorithm. Next, the configuration of the network simulator is introduced and, finally the results obtained using the CCGA, genGA and ssGA are analyzed and compared.

The algorithms have been implemented in Java and tested on a single node for genGA, ssGA and CCGA and on 11 cluster-nodes for dCCGA (distributed CCGA) all nodes having a 3.7 GHz Xeon processor with 16 GB of RAM, running Debian Linux (with kernel 2.6.9-22) and Java version 1.5.0_05.

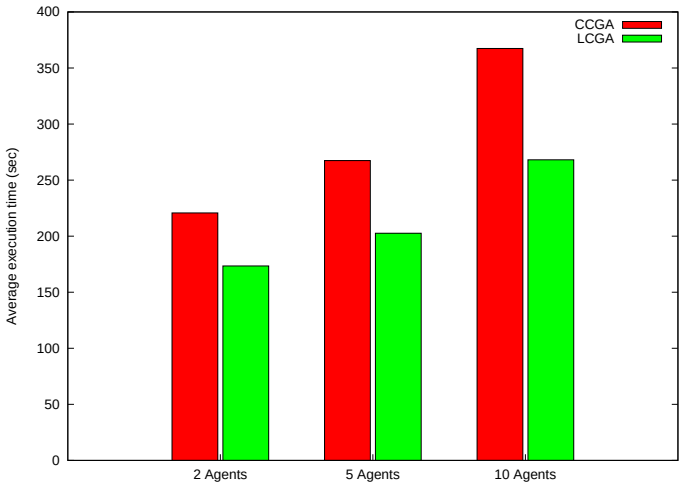


Figure 8.15: Time per experiment for LCGA and CCGA with 2, 5 and 10 subpopulations

8.4.3.1 GA Parameterization

In table 8.10, we show the parameters used for genGA, ssGA, CCGA and dCCGA.

(d)CCGA was tested with 10 subpopulations. For all algorithms we used a randomly generated population composed of 50 individuals. A solution is encoded as a binary string chromosome, in which each gene encodes an integer on 15 bits (see section 8.4.1.1 for a detailed description). Each gene corresponds to one possible bypass link in the half-matrix of all possible links. The selection operator is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). The crossover operator is uniform crossover used with probability p_c=0.8. The mutation operator is bit flip mutation in which each allele of the chromosome is flipped with probability p_m= 1/chromosome_length. Concerning the generational GA and (d)CCGA we have added elitism: the best individual found in one generation is thus kept for the next generation.

Number of Subpopulations	10 (only for (d)CCGA)
(Sub)Population size	50 individuals
Termination Condition	50,000 function evaluations
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual (not for ssGA)

Table 8.7: Parameters used for genGA, ssGA, and (d)CCGA

8.4.3.2 Madhoc Configuration

As for the previous experiments, the Madhoc simulator was used. We chose to analyze and compare the performance of the dCCGA on the same network as presented in 8.4.2.2 (i.e. the 3-clusters network with 42 stations).

8.4.3.3 Results

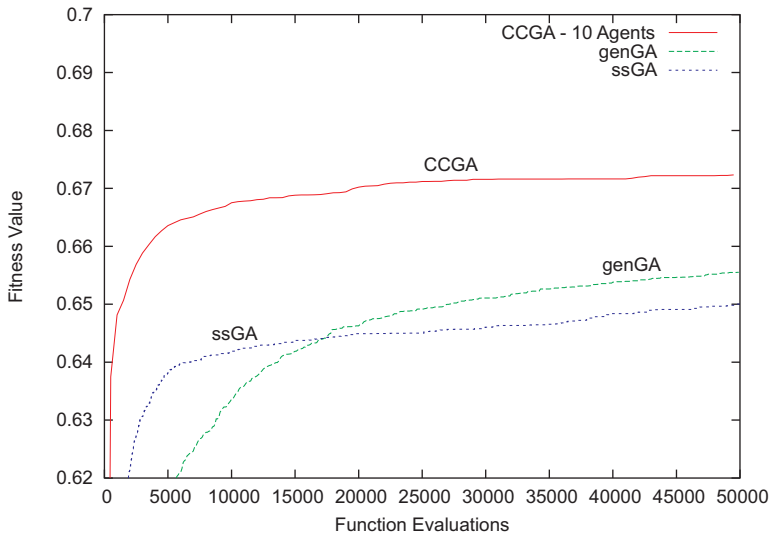


Figure 8.16: Average results of 30 runs using genGA, ssGA and CCGA

Each result presented hereafter is the average obtained on 30 independent runs. In order to establish the statistical significance of the means, we first have checked that the data is normally distributed using the Kolmogorov-Smirnov test. If so, we then perform an ANOVA test so as to compare the means otherwise we use a Kruskal-Wallis test [41].

In Table 8.8 we show the averaged results for all 30 runs for each algorithm.
As it can be seen in Table 8.8, using a CCGA provides better results than both genGA and ssGA,

Network	GA	Crossover	Time	Result
3 Clusters	genGA	Uniform	54min 28s	0.6555
	ssGA	Uniform	58min 24s	0.6500
	CCGA	Uniform	154min 36s	**0.6723**
	dCCGA	Uniform	98min 55s	**0.6723**

Table 8.8: Results of all experiments

genGA being the least performing one in terms of convergence speed and ssGA the least one in terms of best result found (with statistical confidence). This can be graphically observed in Figure 8.16,

as well as the better convergence speed of CCGA compared to the other two GAs. As expected the computational time required for dCCGA is lower than for CCGA thanks to the distribution of the subpopulations, however it is still higher than panmictic GAs like genGA and ssGA due to the synchronization between subpopulation induced by the CCGA algorithm.

8.5 Dynamic Injection Network Optimization

Many real-world optimization problems are dynamic, indeed the objective function, the problem instance or constraints may change over time and thus the optimum of the problem might change as well. If any of these uncertain events have to be taken into account in the optimization process, the problem is called dynamic.

Since EAs are inspired from natural evolution, which is a continuous adaptation process, they appear to be good candidates for dynamic optimization. In these situations, EAs need not only to search for a global optimum, but also to track that optimum over time. This may be trivial if the change in the solution landscape only involves a slight migration of the location of the optimum through the solution space, but if the landscape changes radically the task of the EA becomes much harder. In particular, the problem that EAs face is that, as the population simulated converges on a solution, the diversity within the population diminishes. The population therefore loses its effectiveness at exploring the landscape, and thus the ability to find new, emerging solutions.

Some surveys on EAs applied to dynamic problem can be found in the following three books, [173] from Branke, [174] from Weicker and [175] from Morrison.

In this section, we investigate the use of a CGA on a dynamic optimization problem, i.e. the dynamic injection network.

In section 8.5.1, we first start by presenting the different EAs and some other metaheuristics which have been used to handle dynamic optimization problems. Then in section 8.5.2 we investigate an important new issue which comes with applying EAs for dynamic optimization: performance measurements. Finally, in section 8.5.3, we analyze and compare the performance of a CCGA to two panmictic GAs (generational and steady-state) on a dynamic instance of the injection network problem.

8.5.1 Evolutionary Algorithms for Dynamic Environments

EA is still the largest optimization technique used in dynamic environments. Usually information from the previous search space is simply transferred by keeping individuals in the population. However when dealing with generational EA for dynamic problems, having outdated fitness information

is not an issue.

Branke in [176] proposed a classification in three categories of the main approaches that address this issue of convergence:

- Diversity control

  This category can be divided into two sub-categories since it is possible to increase diversity after a change like with the hypermutation method, or to maintain a high diversity throughout the evolutionary run as in random immigrants or thermodynamical genetic algorithm (TDGA). Hypermutation consists in increasing drastically the mutation rate for a number of generations as soon as a change in the environments is detected. The problem is that increasing diversity is basically equivalent to replacing "good" individuals by random ones and it is difficult to determine the useful amount of diversity (too much makes random walk and too little will not solve the convergence problem). In random immigrants, random individuals are inserted into the population in every generation. The concept behind TDGA is to control explicitly the diversity in the population by controlling its "free energy" (for more information see [177]).

- Memory-based approaches

  Memory-based techniques use an extra memory to store implicitly or explicitly useful information to guide future search. This appears to be very useful when the optimum repeatedly returns to previous locations. Implicit memory simply consists in using a redundant representation and let the EA make a proper use of it. With explicit memory, specific information (that can be solutions) are stored and retrieved by the evolutionary mechanism.

- Multi-populations approaches

  The population is divided into several subpopulations that become specialized in some parts of the search space which allows to track multiple peaks. Those subpopulations maintain information about some promising regions of the search space and thus facilitates the process of tracking the optima as they move. Some examples are the self-organizing scouts (SOS) proposed by Branke in [173] or the multinational GA proposed by Ursem in [178].

Those techniques come with several drawbacks. Indeed, when using hypermutation we destroy one part of the information because of randomization, with random immigrants and TDGA the optimization process gets disturbed and using memory-enhanced EAs appears to be useful only when optimum reappears at old locations and the problem of convergence remains. Due to these restrictions, the use of multi-population approaches appears to be well adapted for dynamic problems while being able to maintain a useful diversity.

Other EAs and other metaheuristics have been applied to dynamic optimization problems, therefore we propose to extend this list with the following algorithms also applied in the literature:

- Cellular Genetic Algorithm

  Alba and Saucedo in [179] demonstrated the good performance of a structured GA (i.e. cellular GA) compared to two panmictic GAs (generational and steady-state) on the dynamic knapsack problem. They have studied the suitability of such algorithms for non-stationary problems, with different severities of changes, using Weicker's metrics (as presented in section 8.5.2).

- Evolution Strategies

  A first rigorous analysis of the performance of a $(1 + 1)$-strategy on a discrete, dynamic objective function has been presented by [180]. Since then, many other research works focused on dynamic optimization problems. The tracking behavior of a $(1 + \lambda)$-strategy is subject of a recent paper by [181] in which a one-max like dynamic problem is considered on a two-dimensional lattice. Another recent application is presented in [182], in which a multiparent evolution strategy ($(\mu/\mu, \lambda)$-ES) employs cumulative step length adaptation for a tracking problem with linear dynamics of the target.

- Ant Colonies

  Ant Colonies Optimization (ACO) has also been used for dynamic problems optimization. Chronologically, we can cite the works of Guntsch and Middendorf [183], in which a population based ACO (P-ACO) is used for optimizing a dynamic TSP (Traveling Salesperson Problem) and a dynamic QAP (Quadratic Assignment Problem. Then Montemanni et al. studied the performance of an ACS (Ant Colony System) on a dynamic vehicle routing problem (DVRP) in [184]. ACS is an element of the family of the ACO family of algorithms (see [185]). Xiao et al. in [186] optimize a dynamic real-world industrial problem (curing of polymer coating) using a Ant Colony System (ACS). Finally, Fernandes et al. in [187] introduce a so-called Binary Ant Algorithm (BAA) which they test on two test problems (Oscillatory Royal Road and Dynamic Schaeffer's function).

- Particle Swarm

  Particle Swarm Optimization (PSO) has already been extensively applied to dynamic problems. Like for other EAs which have to be applied to dynamic environments, PSO must be modified in order to provide optimal results. One critical issue is diversity loss, due to convergence. Therefore, re-diversification mechanisms were introduced using several solutions: randomization [188], repulsion [189], dynamic networks [190] [191], multi-populations [192] [193]. Some other variants were also introduced, like the Unified PSO (UPSO) from Parsopoulos [194] or collaborative models like Lung's Collaborative Evolutionary-Swarm Optimization (CESO) [195].

- Differential evolution

  In [196], Mendes and Mohais introduced a new approach to dynamic optimization problem, using differential evolution (DE). Their algorithm, DynDE, is a multi-population DE that can use dynamic F and CR control parameters and a different scheme for each individual. It is

specifically designed for dynamic environments that only make minor changes - good solutions tend to remain relatively good. They have tested and fine-tuned their DynDE on a well-known benchmark called Moving Peaks, introduced at the same time by Branke in [197] and Morrisson and De Jong in [177].

- Coevolutionary Genetic Algorithm

  In [198], the authors examine the behavior of their Coevolutionary Genetic Algorithm on dynamic environments. This CGA consists of two populations: solution-level one and schema-level one. The solution-level population searches for the good solution in a given problem. The schema-level population searches for the good schemata in the former population. The CGA performs effectively by exchanging genetic information between these populations. They provide some experimental results on the Dynamic Constraint Satisfaction Problems.

- Artificial Immune System

  An Artificial Immune System (AIS) was used by Hart and Ross in [199] on a dynamic scheduling problem.

Through this state of the art, we can conclude that many different EAs and metaheuristics have been used to tackle dynamic optimization problems. However, when focusing on the use of coevolutionary GAs on dynamic problems, we noticed that it is very limited up to now, since only one article studied the utilization of a CGA on a dynamic constraint satisfaction problem (see section 8.5.1).

For this reason, and due to the fact that multi-population approaches appeared to be well adapted maintain a useful diversity, we decided to investigate the use of a CCGA on a dynamic version of the injection network problem.

8.5.2 Performance Measures in Dynamic Environments

An important issue that comes with dynamic optimization problems is the performance measurement. Indeed, a single and time invariant optimum solution does not exist and thus traditional measures of EA performance (best-so-far curves, offline performance, online performance) are inappropriate. Some new measurements have been proposed in order to address this problem :

- Modified offline performance measure, introduced by Branke in [173], where the best-so-far value is reset after each change in the fitness landscape. The problem is that it requires the knowledge of when the fitness landscape changes.

- Difference between the optimum value and the value of the best individual in the environment before a change in the environment. It has the same drawback as the modified offline performance measure.

- Average Euclidean distance to the optimum at each generation. The global optimum in the search space has to be known, which is feasible only in test problems.

- Best-of-generation averages, at each generation, for many EA runs of the same specific problem. This method allows to compare performances at each specific generation but not on the entire range of landscape dynamics. Consequently, it is very difficult to compare experimental results.
- Best-of-generation minus the worst within a small window of recent generations, compared to the best within the window minus the worst within the window. This measurement relies on the strong assumption that the best fitness value will not change much over a small number of generations, which might be not true.
- Collective Mean Fitness [200] consists in the average best-of-generation values averaged over a sufficient number of generations, *G'*, required to expose the EA to a representative sample of all possible landscape dynamics, further averaged over multiple runs. To use this metric it is then necessary to determine the number of generations to use for a representative sample, and this is not straightforward when the landscape dynamics are not known. It has to be established experimentally.
- Weicker introduced three characteristics as performance measures in non-stationary environments in [201]. These are *accuracy* at time *t*, *stability* (an adaptive algorithm is said to be stable if the accuracy is not severely affected by a change in the environment) and *reactivity* (the ability of the adaptive algorithm to react quickly to changes, i.e. to reach again a certain approximation level).

As can be seen, many different performance measures have been developed in order to accurately evaluate and compare EAs performances in dynamic environments. However, when taking a deeper look in the already substantial literature of the domain (cf. the repository of Branke in [202], we can notice that most of the researches have been evaluated with "standard" performance measures. One possible explanation is that the majority of these measures require the knowledge of the optimum and are consequently more targeted to test/benchmark problems than to real-world ones.

8.5.3 Experimental Results

8.5.3.1 GA Parameterization

In table 8.9, we show the parameters used for genGA, ssGA and CCGA.

CCGA was tested with 5 subpopulations. For all algorithms we used a randomly generated population composed of 100 individuals. Both solution encodings as presented in section 8.4.1.1 were used compared. The selection operator is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). Both 2-point and uniform crossover operators were used with probability p_c=1.0. The mutation operator is bit flip mutation in which each allele of the chromosome is flipped with probability p_m= 1/chromosome_length. Concerning the generational GA and CCGA we have added elitism: the best individual found in one generation is thus kept for

the next generation.

Each algorithm was run for 300.000 function evaluations and the problem (i.e. the injection network) was changing after each 50.000 function evaluations. Therefore the different algorithms had to adapt their solutions to the changing problem (i.e. to 6 different snapshots of the mobile injection network as shown in Fig. 8.17).

Number of Subpopulations	5 (only for CCGA)
(Sub)Population size	100 individuals
Termination Condition	300,000 function evaluations
Selection	Binary Tournament
Crossover operator	2-Point and Uniform, p_c=1.0
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual (not for ssGA)

Table 8.9: Parameters used for genGA, ssGA, and (d)CCGA

8.5.3.2 Madhoc Configuration

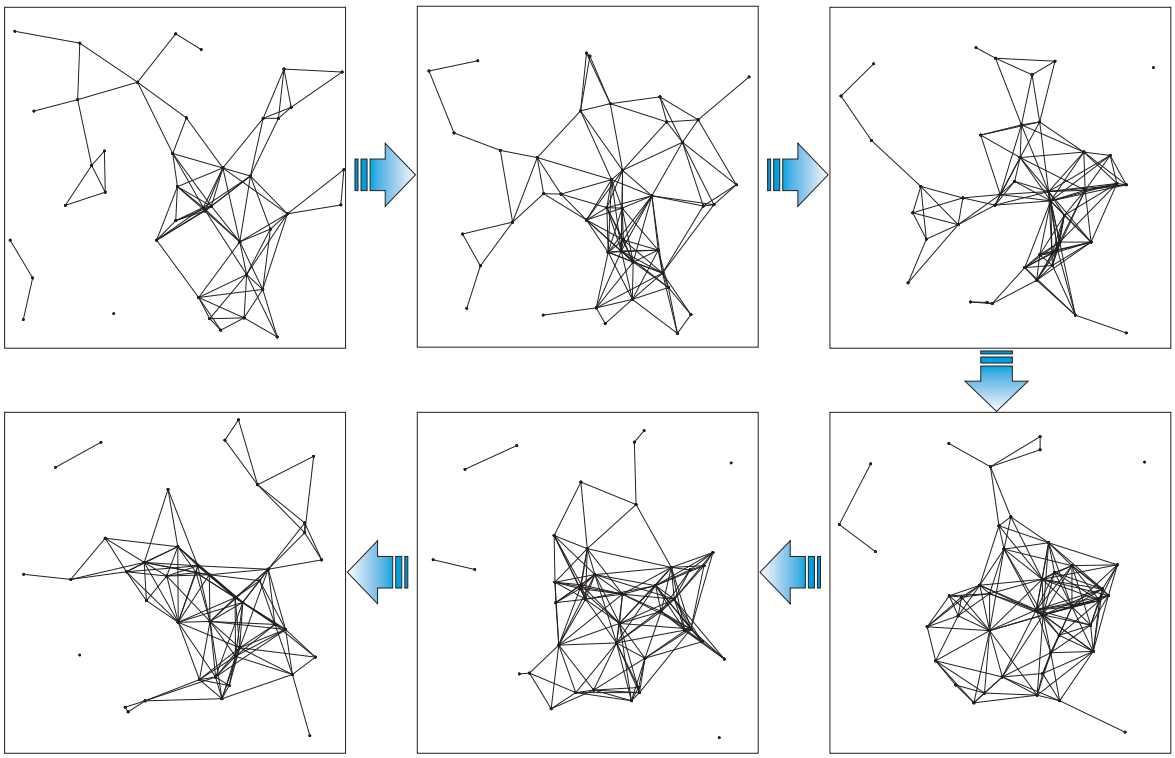


Figure 8.17: Optimized dynamic injection network

As for the previous experiments, the Madhoc simulator was used to simulate the same network as presented in 8.4.2.2 (i.e. the 3-clusters network with 42 stations). In order to conduct our experiments on a dynamic injection network, we used one mobility model provided by Madhoc: the *random waypoint mobility model*. Random waypoint is the most popular mobility model. In the random waypoint mobility model, as described in [203], each node randomly chooses a destination location (in terms of its x, y coordinates) in the simulation area and moves towards this destination with a randomly chosen velocity. When the destination is reached, the station remains at the same place for a while. Once this time expires, the node chooses a random destination in the simulation area and a speed that is uniformly distributed in [minspeed, maxspeed]. The node then travels toward the newly chosen destination at the selected speed. This process is repeated by each station until the end of the simulation.

The velocity of the devices was set between 10 and 50 meters per second and the genetic algorithms were applied on six consecutive snapshots of the mobile network. Each snapshot represents the mobile ad hoc network's state after 15 iterations of the simulator, one iteration representing 0.25 seconds, thus 3.75 seconds. The six different snapshots of the studied mobile injection network are shown in Figure 8.17.

Madhoc parameter	Value
Simulation resolution	0.25 s
Simulation steps	6 times 15 steps
Mobility model	Random waypoint
Velocity	10 m/s $< v <$ 50 m/s

Table 8.10: Parameters used for genGA, ssGA, and (d)CCGA

8.5.3.3 Experimental Results

The results presented hereafter are average out of 30 independent runs. In order to establish the statistical significance of the means, we first have checked that the data is normally distributed using the Kolmogorov-Smirnov test. If so, we then perform an ANOVA test so as to compare the means otherwise we use a Kruskal-Wallis test [41].

Table 8.11 shows the average best results obtained by all the algorithms on each snapshot of the mobile injection network (i.e. after 50.000, 100.000, 150.000, 200.000, 250.000 and 300.000 function evaluations). We can observe that all the best results are obtained by the CCGA, either using the first or the second representation.

When using the first representation, the worst results are always reached by the SGA and the ssGA is always between the SGA and the CCGA, as can be seen in Figure 8.18. With the second

representation the results are significantly changed since the ssGA becomes the least performing algorithm, since its performance is degraded while the SGA's is improved (see Figure 8.19).

Concerning the crossover operators, their influence is the same for the three algorithms. When using the first representation, the results are better with the 2-point crossover, and conversely when using the second representation the best results a obtained with the uniform crossover.

Comparing the computational times. it clearly appears that the multi-population algorithm (i.e. the CCGA) requires much more time than the two panmictic algorithms. In average, using the first representation implies a lower computational time for the SGA and the CCGA. The behavior of the ssGA on this criterion is opposite. The same way, using the 2-point crossover implies a lower computational time for the SGA and the CCGA and the opposite for the ssGA (i.e. a lower computational time with the uniform crossover).

Finally, another important advantage of the CCGA compared to the panmictic GAs is that it shows a better *stability* (cf. Weicker's performance measures) since the average fitness of the best individual is less degraded by a problem change as shown in Figures 8.18 and 8.19. Referring again to Weicker's performance measurement, the CCGA provides the best reactivity, i.e. the CCGA manages to react the fastest the problem changes, in all cases except with the second representation with the 2-point crossover (see Fig 8.19) with which the panmictic GAs provide a better reactivity.

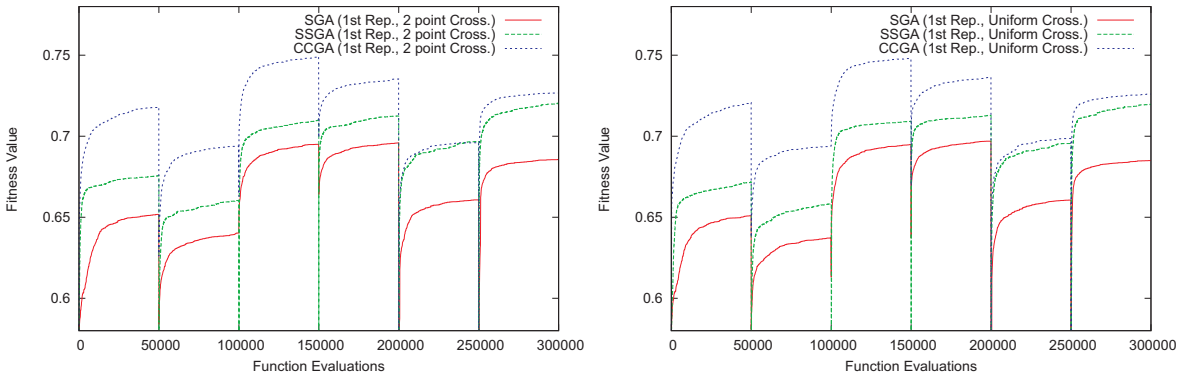


Figure 8.18: Average results of 30 runs using genGA, ssGA and CCGA with the first representation on the dynamic injection network

8.6 Conclusion

Through this chapter we have illustrated the use of our framework and of the CGAs it provides on static and dynamic instances of a new topology control problem dedicated to mobile ad hoc networks called injection network problem.

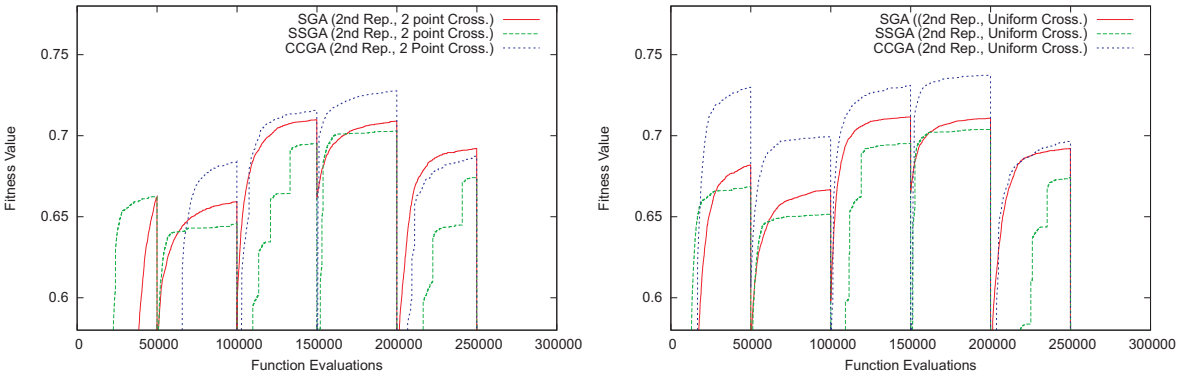


Figure 8.19: Average results of 30 runs using genGA, ssGA and CCGA with the second representation on the dynamic injection network

GA	Repres.	Crossover	Time	Average Best Result After					
				50.000	100.000	150.000	200.000	250.000	300.000
SGA	1st Rep.	DPX	495min 40sec	0.652	0.640	0.695	0.696	0.661	0.686
		UX	385min 31sec	0.651	0.637	0.695	0.697	0.661	0.685
	2nd Rep.	DPX	631min 36sec	0.662	0.659	0.710	0.709	0.692	-0.139
		UX	599min 6sec	0.682	0.667	0.711	0.711	0.692	-0.138
ssGA	1st Rep.	DPX	405min 24sec	0.676	0.660	0.710	0.713	0.697	0.720
		UX	573min 33sec	0.672	0.658	0.709	0.713	0.695	0.720
	2nd Rep.	DPX	418min 44sec	0.661	0.644	0.700	0.702	0.671	-0.200
		UX	445min 40sec	0.668	0.651	0.695	0.704	0.674	-0.110
CCGA	1st Rep.	DPX	1548min 42sec	0.718	0.694	**0.749**	0.735	0.696	**0.727**
		UX	791min 59sec	0.721	0.694	0.748	0.736	**0.699**	0.726
	2nd Rep.	DPX	4260min 25sec	-20.767	0.684	0.716	0.728	0.687	-0.084
		UX	3845min 37sec	**0.730**	**0.699**	0.731	**0.737**	0.696	-0.111

Table 8.11: Results of all experiments

We demonstrated that the use of CGAs is suitable for such a problem, both static and dynamic. Indeed, CCGA and LCGA performed well compared to panmictic algorithms (generational and steady-state GAs) and the best combinations in terms of solution representation and crossover operators were experimentally found.

In order to conduct these experiments, we showed that the DAFO framework can be interfaced with other third-party softwares like the Madhoc simulator we used for simulating our injection networks. We also had the opportunity to make a first evaluation of the distributed version of DAFO (see section 8.4.3).

As current investigations, we work on an efficient decomposition of the problem that can be used by the LCGA. On the one hand this would allow to speed up the calculation process since subpopulations would only evaluate local functions instead of a single global one and on the other hand this would permit a first distribution of the algorithm since no global communication between subpopulations

will be necessary. In order to extend the evaluation the CGAs performances on the static instances, we also started to conduct experiments using a cellular GA.

Concerning the dynamic injection network problem, we planned to evaluate our new LCGAs (dLCGA and hLCGA) on the same problem and on other scenarios.

Another current work consists in defining a multi-objective version of the problem (minimizing the CPL and the number of bypass links, maximizing the clustering coefficient) and to evaluate the results of some state-of-the-art multi-objective GAs such as NSGA-II [204], SPEA2 [205] and MOCell [206].

Part IV

Conclusion and Perspectives

Chapter 9

Conclusion and Perspectives

9.1 Summary

This dissertation has addressed some limitations inherent to the utilization of CGAs and more particulary when applied to static or dynamic real-world problems. Our goal has been to build a multi-agent framework permitting the use, the comparison and the distribution of CGAs on optimization problems with few coding requirements.

To this end, the following issues were tackled:

- Although coevolutionary genetic algorithms (CGAs) are one recent evolution of genetic algorithms (GAs), many different architectures (competitive and cooperative) and applications to function optimization problems have been issued. We brought to the fore that few hybrid and dynamic architectures were proposed as well as few real-world problems were tackled. We additionally demonstrated that CGAs are still rarely usable within the existing platforms/frameworks and that none of them use the agent paradigm to model, implement and distribute their algorithms.

- Therefore we investigated organizational models in multi-agent systems (MAS) as a way to explicitly define the CGAs topologies and the relations/dependencies between their components (i.e. their subpopulations). Due to the development of a new dynamic CGA, we additionally examined reorganizational models to specify the internal dynamics of such a new algorithm and/or its interactions with the environment (i.e. with the optimization problem).

- A new multi-agent model dedicated to evolutionary optimization MAS4EVO (Multi-Agent System for EVolutionary Optimization) was introduced. MAS4EVO provides a novel way of modeling CGAs as organizational and reorganizational multi-agent systems. We demonstrated that based on some organizational primitives, it is possible to define multiple organizations and thus multiple CGAs. Indeed, we modeled two existing CGAs (i.e. CCGA and LCGA) and two new variants of the LCGA,

a hybrid one (hLCGA) and a dynaminc one (dLCGA).

- DAFO (Distributed Agent Framework for Optimization), a multi-agent framework for function optimization using coevolutionary genetic algorithms was presented. DAFO is the implementation of the MAS4EVO model, based on one existing multi-agent platform (i.e. Madkit), which permits to apply, compare and distribute CGAs on optimization problems with a minimum of coding effort. Thanks to the MAS4EVO model, it facilitates the understanding and the manipulation of CGAs structures.

- A stock management problem called Inventory Control Parameter (ICP) problem was chosen as first application scenario of DAFO. We experimentally demonstrated that decomposing the problem in terms of stock items with LCGA provides better results than the CCGA on small instances of this problem but its performance degrades as the problem size increases. We additionally applied the two new variants of LCGA, respectively hybrid (hLCGA) and dynamic (dLCGA), with which we improved the "standard" LCGA's performances on this business problem.

- As second application scenario, we tackled an emergent business problem related to topology control in mobile hybrid ad hoc networks called injection network problem. The objective is to optimize the placement and the number of long-range links, using small-world properties as indicator of a good solution, so as to unpartition ad hoc networks. We demonstrated that the use of our framework and of the CGAs it provides is suitable for both static and dynamic instances of this new problem. Indeed, CCGA and LCGA performed well compared to panmictic algorithms (generational and steady-state GAs) and the best combinations in terms of solution representation and crossover operators were experimentally found. In order to conduct these experiments, we showed that the DAFO framework can be interfaced with other third-party softwares like the Madhoc simulator we used for simulating our injection networks. We also had the opportunity to make a first evaluation of the distributed version of DAFO.

9.2 Future Research

Throughout this dissertation, a number of possible directions for future research have been suggested. In the following, we expand these ideas and provide some additional ones.

Concerning the DAFO framework, thanks to its organizational model and its agents' architecture, it is rather easy to add new solvers, in addition to the generational GAs and the local search algorithms. Therefore, we plan in our future works to provide the possibility to use steady-state and cellular GAs so as to build new CGAs variants.

We also project to have a multiobjective (MO) version of these evolutionary solvers, since multiobjective CGAs are a recent and promising evolution of CGAs [207].

Another extension concerns the interaction protocols that we plan to extend with asynchronous communication between the solvers. This will prevent the solvers to wait for some missing solution pieces from other solver(s), and thus reduce the computational time.

Some other current and future works focus on the injection network optimization problem.

First of all, in order to evaluate the CGAs performances on the static instances to state-of-the art algorithms, we started conducting experiments with an adaptive cellular GA.

Another current work consists in defining a multi-objective version of the problem (minimizing the CPL and the number of bypass links, maximizing the clustering coefficient) and to evaluate the results of some state-of-the-art multiobjective GAs such as NSGA-II [204], SPEA2 [205] and MOCell [206]. Once these first results obtained, as previously mentioned we will integrate this MO algorithm as a new agent solver in DAFO and evaluate this new MO-CGA. These two research works are done in collaboration with the university of Malaga.

In the near future we plan to work on an efficient decomposition of the problem that can be used by the LCGA. Indeed, contrary to the first problem tackled in this dissertation (the ICP problem), no efficient problem decomposition has been found yet. The objective is to speed up the calculation process since subpopulations will only evaluate local functions instead of a single global one and on to permit a first distribution of the algorithm since no global communication between subpopulations will be necessary.

Concerning the dynamic instance of this injection network problem, we planned to evaluate our new LCGAs (dLCGA and hLCGA) on the same problem and on other scenarios. At the same time, some more investigations will have to be done on performance measurements dynamic environments, since as depicted in this dissertation, at the present time they are still targeted to test problems/benchmarks and not to real-world problems.

Part V

Appendix

Appendix A

Multi-Agent Platforms

A.1 Introduction

In this section we present different multi-agent platforms, the advantages/disadvantages they have in the context of function optimization problems and finally in the conclusion we present the one we choose.

In order to find a multi-agent platform which fits best to our needs, we thus investigated several multi-agent platforms amongst the many platforms available.

The emergence of the agent paradigm came with several architectures and methodologies to model multi-agent systems. Here are some of them: MaSE (multi-agents software engineering) [208], AGR (agent group role) [124], Gaia [209] and many others. Based on this modeling theory, many platforms are currently available on the market but only a few appear to be relevant and popular, that's why we only focused on the following: JADE [31], Zeus [210], AgentTool [211], AgentBuilder, Jack [212] and Madkit [213].

With more than sixty different multi-agent platforms available, finding the adapted platform is quite a complex task. In order to compare them, it is important to notice that most of those platforms are based on a particular agent model (e.g. Zeus) or targeted to a specific domain, such as simulation (Swarm) or mobile agents (Aglets). It is then clear that simulation tools won't be adapted to our requirements as well as platforms which use agent models that are not organizational models.

Additionally, the current literature provides only a few papers targeted to agent platforms evaluation and comparison among which the following ones appeared to be relevant: , [214], [215], the last one being from Leszczyna in [216] in 2004.

According to the information related in these papers, we decided to investigate the following toolkits:

A.2 Zeus

Zeus is a well-known open source multi-agent toolkit developed in Java by the British Telecommunications Laboratory (BT). Zeus is a complete environment using a methodology named ”role modeling” for the development of collaborative systems. Each ZEUS agent consists of a definition layer, an organizational layer and a coordination layer. The definition layer comprises the agent's reasoning (and learning) abilities, its goals, resources, skills, beliefs, preferences, etc. The organization layer describes the agent's relationships with other agents, and at the coordination layer the agent is modeled as a social entity, i.e. in terms of the coordination and negotiation techniques it possesses. Zeus has three main functional components: the agent component library, agent building tools and visualization tools. The agent component library is a collection of software components that implement the functionality necessary for multi-agent systems. It provides a set of pre-written and pre-tested agent components. The agent building tools provide an environment for developing agents (including Java code generator) and the visualization tools is a run-time environment for running, testing and debugging agents. The Problem with Zeus is that it seems to be well adapted to developers who lack a strong Java background, however it is quite complex and using it requires a lot of time. Additionally there has been no update since version 1.2.1 of May 2001.

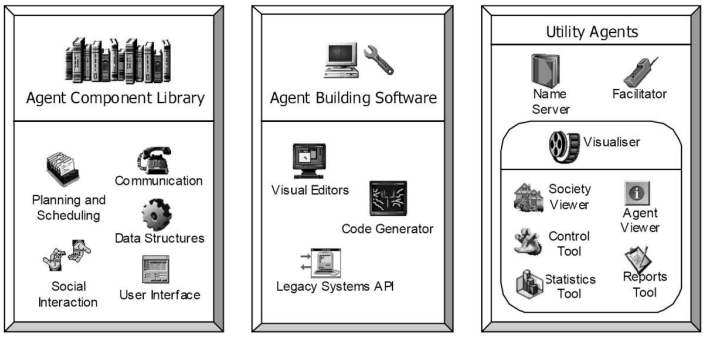


Figure A.1: Components of the Zeus agent building toolkit

A.3 AgentTool

AgentTool is based on the MaSE methodology. Multiagent Systems Engineering (MaSE) is an agent-oriented software engineering methodology which is an extension of the object-oriented approach. MaSE does not view agents as being necessarily autonomous, proactive, etc.; rather agents are ”simple software processes that interact with each other to meet an overall system goal”. MaSE fully

describes the process which guides a system developer from an initial system specification to system implementation. This process consists of seven steps, divided into two phases. The Analysis phase consists of three steps: Capturing Goals, Applying Use Cases, and Refining Roles. The remaining four process steps, Creating Agent Classes, Constructing Conversations, Assembling Agent Classes, and System Design, form the Design phase. MaSE has extensive tool support in the form of AgentTool. Since version 2.03, it implements all seven steps of MaSE. It also provides automated support for transforming analysis models into design constructs. This tool is interesting to carry out the first steps in the development of a MAS which can be interesting but it's not the main objective in this project.

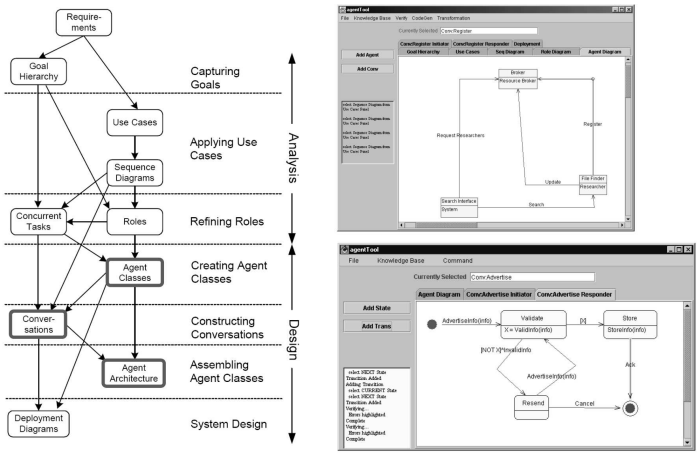


Figure A.2: AgentTool implemented MaSE features, class diagram and conversation diagram

A.4 AgentBuilder

AgentBuilder is an integrated commercial tool suite for constructing intelligent software agents. It is developed by Reticular Systems Inc., and is grounded on the Agent0 (Shoham 93) and Placa (Thomas 93) BDI models. It is available in two versions: AgentBuilder Lite and AgentBuilder Pro. Academic versions are also available. Currently, license fees ranges from $100 to 5000$ depending on the version. AgentBuilder provides a powerful mental model for its agents, allowing developers to easily specify such things as beliefs, intentions, commitments, and behavioral rules. In addition, use is made of popularly accepted standards like KQML for the agent communication language along with use of UML-like object modeling facilities to model domain knowledge of an agent. This tool is remarkable both by the high quality of its software and the well-known academic background model used. However it is a very complex tool which requires many learning efforts and a good knowledge in the MAS domain to be used extensively. It is also quite limited in terms of extensibility, deployment and reusability. And last but not least, it is not a free or open-source tool which is not suitable for the project.

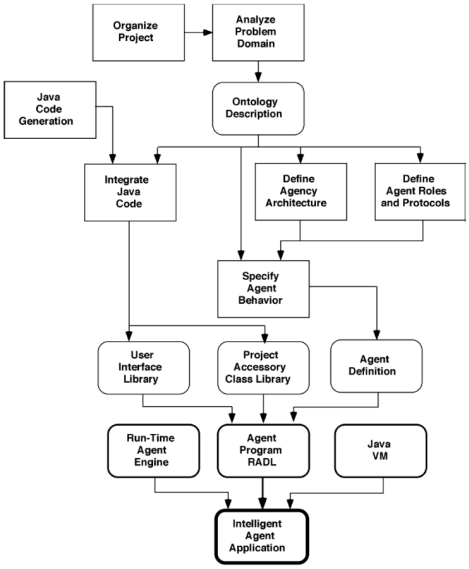


Figure A.3: AgentBuilder: Agent construction process

A.5 Jack

Jack Intelligent Agents is a development environment that is built on top of Java and acts as an extension of Java that offers classes for implementing agent behavior. It is developed by Agent Oriented Software Group, an Australian commercial company. Although it is a commercial tool, a 60-days evaluation license is available. Jack defines its agents as being 'intelligent'. These types of agents model reasoning behavior according to the theoretical Belief-Desire-Intention (BDI) model. The toolkit consists of the JDE (Jack Development Environment), a graphical tool to manage projects, the Jack Agent Language (JAL) compiler, that translates JAL programs to pure Java programs, and a library of supporting classes, called the Jack Agent Kernel. The JAL is an extension of Java. Jack proposes no methodology and focuses mainly on the development stage. Analysis and design are only mentioned in a few and very technical documentations what makes reuse difficult. Jack seems quite hard to use since it is necessary to learn the Jack Agent Language (JAL) and to know the BDI model. Moreover the lack of graphical tools (there is no editor for the development or the deployment) makes the development and deployment of systems difficult. Like AgentBuilder it is a commercial tool and thus it is not appropriate.

A.6 Jade

Jade has been conceived and developed by Tilab, the Telecom Italia R&D center. It is a free and open source software implemented in Java. It provides a container paradigm for associating JVM's, an agent foundation class for writing customized agents, a library of protocol skeletons and an interface

Figure A.4: Jack Intelligent Agents Components

for using the JESS rule-based system for the behavior of the agents. Jade can be distributed over several hosts, each host runs a container connected to the main container of the platform. Distributed agents can then transparently communicate and can even clone or migrate on different containers Since JADE version 3.0b1 LEAP libraries (Lightweight Extensible Agent Platform) are completely integrated, providing a runtime environment that can be deployed on a wide range of devices varying from servers to Java enabled cell phones with J2ME MIDP. There is no methodology specified for the development but it comes with tools that support the debugging and the deployment phases. However there is no interface for the development or the implementation, and consequently the implementation phase requires a lot of efforts. This tool has already been used in another small project and this experience plus the other preceding remarks show that Jade won't suit to such a project.

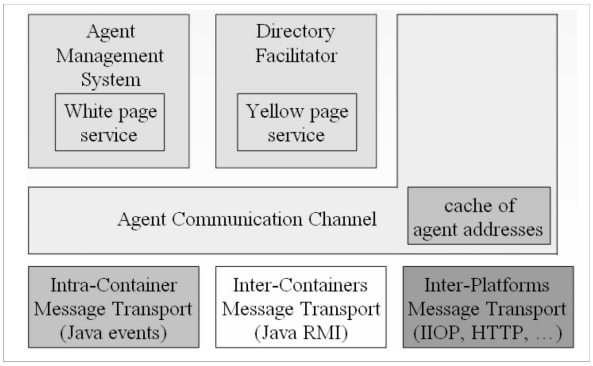


Figure A.5: Jade architecture

A.7 Madkit

MadKit is a Java multi-agent platform built upon the Agent/Group/Role (AGR) organizational model. It is developed by Gutknecht and Ferber at the LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier), a public research laboratory. Madkit is a free software which comes with a mix of GPL/LPGL licenses (LGPL for the basic libraries, and GPL for development tools). Madkit does not enforce any consideration about the internal structure of agents and thus allows the developer to freely implement its own agent architectures. MadKit is also a distributed platform which allows the development of efficient distributed applications thanks to its micro-kernel architecture. All considerations about basic distributed components such as "sockets" and "ports", are totally transparent and contrary to many other frameworks the MadKit distribution mechanisms do not use the quite rather slow techniques of RMI or CORBA remote access. The main disadvantage of Madkit is that building complex agents requires a lot of code since there is no pre-defined agent model, however this adds flexibility.

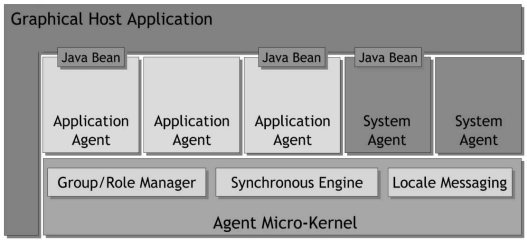


Figure A.6: Madkit architecture

Based on this state of the art, only two platforms still match our requirements: Jade and Madkit. Indeed, they both are open source softwares, they allow the use of an organizational model and they can be distributed. Madkit has been our final choice, because of the possibility of using the group and roles already implemented for AGR model.

A.8 Performance Evaluation

Those few papers related to the topic only compare the platforms in a qualitative point of view. However, since our multi-agent system is targeted to function optimization, its performance and thus the performance of the multi-agent platform is also a key issue. However, until 2004 there was no paper providing such a performance comparison between agent platforms, this lack was outlined in [217].

Only three recent papers propose a performance comparison of the message transport systems of some agent toolkits. Jun in [218] compares Jade, Zeus and Jack, Burbeck in [219] compares Jade, Tryllian and SAP and finally Mulet [30] compares Jade, Madkit and AgentScape. Mulet also extends

the comparison to Directory Services.

According to Mulet, implementing basic services, like messaging, by means of agents as opposed to a kernel implementation allows a high modularity but degrades performance. This is why Madkit is slightly slower than Jade but still much more performant than AgentScape. The situtation is reversed concerning the service directory service, which is offered directly by Madkit and by means of agent by Jade. Additionally, Madkit distributes and duplicates its directories among all the kernels, which make service discovery in a distributed mode even faster.

Madkit is thus close to Jade in terms of performance, which appears to be a good point since Jade also provides good performances compared to other platforms (Zeus, Jack, Tryllian and SAP).

Appendix B

DAFODL's DTD

```
 <?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT evoframework (interactiongraph,geneticparameters, localsearchparameters*)>
<!ELEMENT interactiongraph (topology, numberofagents, exchangedinformation)>
<!ELEMENT topology (#PCDATA)>
<!ELEMENT numberofagents (#PCDATA)>
<!ELEMENT exchangedinformation (#PCDATA)>
<!ELEMENT geneticparameters (algorithm, fitnessclass, experiments, terminationcondi-
tion, terminationconditionvalue, numchroms, numgenes, sizegenes, crossover, crossrate,
mutrate, elitenumber, localsearch*)>
<!ELEMENT algorithm (#PCDATA)>
<!ELEMENT fitnessclass (#PCDATA)>
<!ELEMENT experiments (#PCDATA)>
<!ELEMENT terminationconditionvalue (#PCDATA)>
<!ELEMENT numchroms (#PCDATA)>
<!ELEMENT numgenes (#PCDATA)>
<!ELEMENT sizegenes (#PCDATA)>
<!ELEMENT crossover (#PCDATA)>
<!ELEMENT crossrate (#PCDATA)>
<!ELEMENT mutrate (#PCDATA)>
<!ELEMENT elitenumber (#PCDATA)>
<!ELEMENT localsearch (#PCDATA)>
<!ELEMENT localsearchparameters (lsalgorithm, lsexchangedinformation, lspopulationrate*,
```

```
lsterminationcondiation)>
<!ELEMENT lsalgorithm (#PCDATA)>
<!ELEMENT lsexchangedinformation (#PCDATA)>
<!ELEMENT lspopulationrate (#PCDATA)>
<!ELEMENT lsterminationcondition (#PCDATA)>
```

Bibliography

[1] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity (USA)*, vol. 4, no. 4, pp. 31–52, 1999. xi, 4, 5, 36, 41, 42, 45

[2] O. Boissier, *Principes et architecture des systèmes multi-agents*. Paris, France: Hermès Science Publications, 2001, ch. Modèles et architectures d'agents. xvi, 66

[3] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, 1997. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=585893 3, 36

[4] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20{\&}path=ASIN/0472084607 4, 35

[5] H. Cobb and J. Grefenstette, "GA for Tracking Changing Environments," in *PPSN*, 1993, pp. 532–530. 4, 37

[6] D. Whitley and J. Kauth, "GENITOR: A Different Genetic Algorithm," in *Proceedings of the Rocky Mountain Colorado, on Artificial Intelligence*, 1988, pp. 118–130. 4, 38, 186

[7] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms." *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002. 4, 42, 45, 57

[8] Z. Konfrst, "Parallel genetic algorithms: Advances, computing trends, applications and perspectives," in *IPDPS*. IEEE Computer Society, 2004. 4, 5, 42, 45

[9] K. A. DeJong, "An analysis of the behaviour of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, Michigan, 1975. 5, 169

[10] J. Paredis, "Coevolutionary life-time learning," in *Parallel Problem Solving from Nature – PPSN IV*. Berlin: Springer, 1996, pp. 72–80. 5, 46

[11] M. A. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature – PPSN III*. Berlin: Springer, 1994, pp. 249–257. 6, 7, 49, 50, 51, 186

[12] F. Seredynski, "Loosely coupled distributed genetic algorithms," in *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1994, pp. 514–523. 6, 48

[13] ——, "Competitive coevolutionary multi-agent systems: the application to mapping and scheduling problems," *J. Parallel Distrib. Comput.*, vol. 47, no. 1, pp. 39–57, 1997. 7, 48, 54

[14] R. Eriksson and B. Olsson, "Cooperative coevolution in inventory control optimisation," in *Proc. of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*. University of East Anglia, Norwich, UK: Springer-Verlag, 1997. 7, 21, 50, 162

[15] J. B. Pollack and A. D. Blair, "Coevolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, pp. 225–240, 1998. 7, 47

[16] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 264–270. 7, 48

[17] M. A. Potter, L. Meeden, and A. C. Schultz, "Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists," in *IJCAI*, 2001, pp. 1337–1343. [Online]. Available: citeseer.ist.psu.edu/article/potter01heterogeneity.html 7, 49

[18] S. Cahon, N. Melab, and E.-G. Talbi, "ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004. 7, 58

[19] E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa, "MALLBA: A library of skeletons for combinatorial optimisation," in *Euro-Par 2002 Parallel Processing*, ser. Lecture Notes in Computer Science, B. Monien and R. Feldman, Eds. Berlin Heidelberg: Springer-Verlag, 2002, vol. 2400, pp. 927–932. [Online]. Available: citeseer.ist.psu.edu/678882.html 7, 59

[20] M. G. Arenas, P. Collet, A. E. Eiben, M. Jelasity, J. J. Merelo, B. Paechter, M. Preuß, and M. Schoenauer, "A framework for distributed evolutionary algorithms," in *Parallel Problem Solving from Nature - PPSN VII*, ser. Lecture Notes in Computer Science, J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas, and H.-P. Schwefel, Eds., vol. 2439. Springer-Verlag, 2002, pp. 665–675. [Online]. Available: citeseer.ist.psu.edu/arenas02framework.html 7, 58

[21] S. Luke, "Ecj: A java evolutionary computation library," 2006. [Online]. Available: http://cs.gmu.edu/$\sim$eclab/projects/ecj/ 7, 59

[22] B. Bauer, J. Muller, and J. Odell, "Agent UML: A formalism for specifying multiagent interaction," Berlin, Germany, pp. 91–103, 2001. [Online]. Available: citeseer.ist.psu.edu/448042.html 13, 109, 110

[23] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: An organizational view of multi-agent systems." in *AOSE*, ser. Lecture Notes in Computer Science, P. Giorgini, J. P. Müller, and J. Odell, Eds., vol. 2935. Springer, 2003, pp. 214–230. 13, 70, 71, 109, 110

[24] B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois, "MOISEInst: An organizational model for specifying rights and duties of autonomous agents." in *EUMAS*, M. P. Gleizes, G. A. Kaminka, A. Nowé, S. Ossowski, K. Tuyls, and K. Verbeeck, Eds. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 2005, pp. 484–485. [Online]. Available: http://dblp.uni-trier.de/db/conf/eumas/eumas2005.html#GateauBKD05 14, 71, 73, 100

[25] Y. S. Son and R. Baldick, "Hybrid coevolutionary programming for nash equilibrium search in games with local optima." *IEEE Trans. Evolutionary Computation*, vol. 8, no. 4, pp. 305–315, 2004. 17, 53, 131

[26] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000. [Online]. Available: citeseer.ist.psu.edu/article/potter00cooperative.html 17, 53, 138

[27] A. Iorio and X. Li, "Parameter control within a co-operative co-evolutionary genetic algorithm," in *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 2002, pp. 247–256. 17, 53, 138

[28] Z. Cai and Z. Peng, "Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems," *J. Intell. Robotics Syst.*, vol. 33, no. 1, pp. 61–71, 2002. 17, 53, 138

[29] O. Gutknecht and J. Ferber, "Madkit: a generic multi-agent platform," in *Proc. of the fourth international conference on Autonomous agents*. ACM Press, 2000, pp. 78–79. 19, 154

[30] L. Mulet, J. M. Such, and J. M. Alberola, "Performance evaluation of open-source multiagent platforms," in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2006, pp. 1107–1109. 19, 222

[31] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with JADE," in *ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*. London, UK: Springer-Verlag, 2001, pp. 89–103. 19, 217

[32] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "Simulating Realistic Mobility Models for Large Heterogeneous MANETS," in *Demo proceeding of the 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'06)*. IEEE, October 2006. 24, 185

[33] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs (2nd, extended ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1994. 37, 180, 186

[34] E. Alba, A. J. Nebro, and J. M. Troya, "Heterogeneous computing and parallel genetic algorithms," *J. Parallel Distrib. Comput.*, vol. 62, no. 9, pp. 1362–1385, 2002. 37, 186

[35] G. Syswerda, "Uniform Crossover in Genetic Algorithms," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9. 40

[36] W. E. Hart, S. B. Baden, R. K. Belew, and S. R. Kohn, "Analysis of the numerical effects of parallelism on a parallel genetic algorithm," in *IPPS '96: Proceedings of the 10th International Parallel Processing Symposium*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 606–612. 41

[37] E. Alba, "Parallel evolutionary algorithms can achieve super-linear performance," *Inf. Process. Lett.*, vol. 82, no. 1, pp. 7–13, 2002. 42

[38] E. Cantú-Paz, "A survey of parallel genetic algorithms," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Tech. Rep. 97003, 1997. 42, 44, 45

[39] E. Alba and J. M. Troya, "Improving flexibility and efficiency by adding parallelism to genetic algorithms," *Statistics and Computing*, vol. 12, no. 2, pp. 91–114, 2002. 43

[40] B. Dorronsoro, E. Alba, M. Giacobini, and M. Tomassini, "The influence of grid shape and asynchronicity on cellular evolutionary algorithms," in *IEEE International Conference on Evolutionary Computation*, Y. Shi, Ed. Portland, Oregon: IEEE Press, June 20–23 2004, pp. 2152–2158. [Online]. Available: citeseer.ist.psu.edu/dorronsoro04influence.html 43

[41] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005. 43, 190, 197, 201, 208

[42] P. B. Grosso, "Computer simulations of genetic adaptation: parallel subcomponent interaction in a multilocus model," Ph.D. dissertation, Ann Arbor, MI, USA, 1985. 44

[43] E. Cantú-Paz, "Migration policies, selection pressure, and parallel evolutionary algorithms," *Journal of Heuristics*, vol. 7, no. 4, pp. 311–334, 2001. 44

[44] E. Cantú-Paz, "Topologies, migration rates, and multi-population parallel genetic algorithms," in *GECCO*, W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 91–98. 44

[45] E. Cantú-Paz and D. E. Goldberg, "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 221–238, June 2000. [Online]. Available: http://www.sciencedirect.com/science/article/B6V29-40CRYKF-6/2/9ec1963003d8d255648b3d7878df627245

[46] Z. Skolicki and K. D. Jong, "The influence of migration sizes and intervals on island models," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 1295–1302. 45

[47] F. Gruau, "Neural network synthesis using cellular encoding and the genetic algorithm." Ph.D. dissertation, France, 1994. [Online]. Available: citeseer.ist.psu.edu/frederic94neural.html 45

[48] M. Gorges-Schleuter, "Asparagos96 and the traveling salesman problem," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, 1997, pp. 171–174. 45

[49] S.-C. Lin, E. D. Goodman, and I. William F. Punch, "Investigating parallel genetic algorithms on job shop scheduling problems," in *EP '97: Proceedings of the 6th International Conference on Evolutionary Programming VI*. London, UK: Springer-Verlag, 1997, pp. 383–393. 45

[50] E. Alba and J. M. Troya, "Influence of the migration policy in parallel distributedgas with structured and panmictic populations," *Applied Intelligence*, vol. 12, no. 3, pp. 163–181, 2000. 45

[51] ——, "Analyzing synchronous and asynchronous parallel distributed genetic algorithms," *Future Gener. Comput. Syst.*, vol. 17, no. 4, pp. 451–465, 2001. 45

[52] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. 45, 55, 186

[53] R. Bianchini and C. Brown, "Parallel genetic algorithms on distributed-memory architectures," in *NATUG-6: Proceedings of the sixth conference of the North American Transputer Users Group on Transputer research and applications 6*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 1993, pp. 67–82. 45

[54] P. R. Ehrlich and P. H. Raven, "Butterflies and plants: A study in coevolution," *Evolution*, vol. 18, no. 4, pp. 586–608, 1964. 46

[55] C. Darwin, *The Origin of Species by Means of Natural Selection*. NY: Mentor Reprint, 1958, 1859. 46

[56] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," pp. 228–234, 1991. 47

[57] J. Paredis, "Co-evolutionary constraint satisfaction." in *PPSN*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., vol. 866. Springer, 1994, pp. 46–55. 47

[58] ——, "Steps towards co-evolutionary classification neural networks," in *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994, pp. 102–108. 47

[59] H. Juille and J. B. Pollack, "Co-evolving intertwined spirals," in *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, L. J. Fogel, P. J. Angeline, and T. Baeck, Eds. MIT Press, 1996, pp. 461–467. [Online]. Available: citeseer.ist.psu.edu/juille96coevolving.html 47

[60] H. A. Mayer, "Symbiotic coevolution of artificial neural networks and training data sets," *Lecture Notes in Computer Science*, vol. 1498, pp. 511–520, 1998. [Online]. Available: citeseer.ist.psu.edu/mayer98symbiotic.html 47

[61] C. D. Rosin and R. K. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, L. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 373–380. [Online]. Available: citeseer.ist.psu.edu/rosin95methods.html 47

[62] ——, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1–29, 1997. [Online]. Available: citeseer.ist.psu.edu/rosin96new.html 47

[63] D. Schlierkamp-Voosen and H. Mühlenbein, "Strategy adaptation by competing subpopulations," in *Parallel Problem Solving from Nature – PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin: Springer, 1994, pp. 199–208. [Online]. Available: citeseer.ist.psu.edu/schlierkamp-voosen94strategy.html 48, 50

[64] K. Sims, "Evolving 3d morphology and behavior by competition," *Artif. Life*, vol. 1, no. 4, pp. 353–372, 1994. 48

[65] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," in *Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997. [Online]. Available: citeseer.ist.psu.edu/luke97coevolving.html 48

[66] D. Floreano and S. Nolfi, "Adaptive behavior in competing co-evolving species," in *Fourth European Conference on Artificial Life*, P. Husbands and I. Harvey, Eds. Cambridge, MA: The MIT Press, 1997, pp. 378–387. [Online]. Available: citeseer.ist.psu.edu/floreano97adaptive.html 48

[67] F. Seredynski, J. Koronacki, and C. Z. Janikow, "Distributed scheduling with decomposed optimization criterion: Genetic programming approach," in *Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*. London, UK: Springer-Verlag, 1999, pp. 192–200. 48, 56

[68] F. Seredynski, A. Y. Zomaya, and P. Bouvry, "Function optimization with coevolutionary algorithms," in *Proc. of the International Intelligent Information Processing and Web Mining Conference*. Poland: Springer, 2003. 48, 53, 177

[69] P. Husbands and F. Mill, "Simulated co-evolution as the mechanism for emergent planning and scheduling," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufman, 1991, pp. 264–270. 49

[70] M. A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, 1997. 49, 53

[71] M. A. Potter and K. A. D. Jong, "The coevolution of antibodies for concept learning," in *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1998, pp. 530–539. 49

[72] M. A. Potter, K. A. D. Jong, and J. J. Grefenstette, "A coevolutionary approach to learning sequential decision rules," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 366–372. 49

[73] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000. 49

[74] J. Paredis, "The symbiotic evolution of solutions and their representations," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 359–365. 49

[75] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evolutionary Computation*, vol. 5, no. 4, pp. 373–399, 1997. [Online]. Available: citeseer.ist.psu.edu/moriarty98forming.html 49

[76] R. P. Wiegand, "Applying diffusion to a cooperative coevolutionary model," in *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1998, pp. 560–572. 50

[77] S. Cahon, N. Melab, and E.-G. Talbi, "Building with paradisEO reusable parallel and distributed evolutionary algorithms," *Parallel Comput.*, vol. 30, no. 5-6, pp. 677–697, 2004. 57

[78] N. L. Costa, J. and P. Silva, "Jdeal: The java distributed evolutionary algorithms library," 2008. [Online]. Available: http://www.laseeb.org/sw/JDEAL/ 59

[79] E. Noda, A. L. V. Coelho, I. L. M. Ricarte, A. Yamakami, and A. A. Freitas, "Devising adaptive migration policies for cooperative distributed genetic algorithms," in *Proc. 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*. IEEE Press, 2002. [Online]. Available: http://www.cs.kent.ac.uk/people/staff/aaf/my-publications-ukc.html 59, 60

[80] A. Roli, "Metaheuristics and structure in satisfiability problems," University of Bologna (Italy), Tech. Rep. DEIS-LIA-03-005, May 2003, phD Thesis - LIA Series no. 66. 59

[81] M. Milano and A. Roli, "Magma: A multiagent architecture for metaheuristics," *IEEE Trans. on Systems, Man and Cybernetics – Part B*, vol. 34, no. 2, pp. 925–941, April 2004. 59

[82] S. Cahon, "ParadisEO : Une plate-forme pour la conception et le déploiement de métaheuristiques parallèles hybrides sur clusters et grilles," Ph.D. dissertation, University of Sciences and Technology of Lille, France, July 2005. 61

[83] M. J. Wooldridge and N. R. Jennings, "Agent theories, architectures, and languages: A survey," in *Workshop on Agent Theories, Architectures and Languages (ECAI'94)*, M. J. Wooldridge and N. R. Jennings, Eds., vol. 890. Springer-Verlag, 1995, pp. 1–22. 64

[84] J.-P. Briot and Y. Demazeau, *Principes et architectures des systèmes multi-agents*. Paris, France: Hermès Science Publications, 2001, ch. Introduction aux agents, pp. 17–25. 65

[85] P. D. O'Brien and R. C. Nicol, "FIPA towards a standard for software agents," *BT Technology Journal*, vol. 16, no. 3, pp. 51–59, 1998. 65

[86] Y. Demazeau, "From interactions to collective behaviour in agent-based systems," 1995. [Online]. Available: citeseer.ist.psu.edu/demazeau95from.html 65, 90

[87] J. Ferber, *Les Systèmes multi-agents: Vers une intelligence collective*. Dunod, January 2007. [Online]. Available: http://www.amazon.fr/exec/obidos/redirect?tag=citeulike06-21\&path=ASIN/2729606653 66

[88] M. Wooldridge, *Intelligent agents*. Cambridge, MA, USA: MIT Press, 1999, pp. 27–77. 66

[89] J. Bryson, "Cross-paradigm analysis of autonomous agent architecture," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 12, no. 2, pp. 165–190, 2000. [Online]. Available: citeseer.ist.psu.edu/bryson00crossparadigm.html 66

[90] R. A. Brooks and J. H. Connell, "Asynchronous distributed control system for a mobile robot," in *SPIE's Cambridge Symposium on Optical and Optoelecronic Engineering*, Cambridge, MA, 1986, pp. 77–84. 66

[91] A. Drogoul, B. Corbara, and S. Lalande, "MANTA: New experimental results on the emergence of (artificial) ant societies," in *Artificial Societies: The Computer Simulation of Social Life*, N. Gilbert and R. Conte, Eds. UCL Press: London, 1995, pp. 190–211. [Online]. Available: citeseer.ist.psu.edu/drogoul95manta.html 66

[92] M. P. Georgeff and A. L. Lansky, "Reactive reasoning and planning." in *AAAI*, 1987, pp. 677–682. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaai/aaai87.html#GeorgeffL87 66

[93] I. A. Ferguson, "Integrating models and behaviors in autonomous agents," in *Proceedings of the AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Stanford University, 1995, pp. 78–91. 66

[94] T. Bouron and A. Collinot, "SAM: a model to design computational social agents," in *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*. New York, NY, USA: John Wiley & Sons, Inc., 1992, pp. 239–243. 66

[95] Y. Shoham, "Agent oriented programming," *Artificial Intelligence*, vol. 60, pp. 51–92, 1993. 66

[96] J. P. Muller, *The Design of Intelligent Agents: A Layered Approach*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. 66

[97] F. Dignum, D. Kinny, and L. Sonenberg, "Motivational attitudes of agents: On desires, obligations, and norms," in *CEEMAS '01: Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*. London, UK: Springer-Verlag, 2002, pp. 83–92. 66

[98] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0137903952 67

[99] J. L. Austin, *How to do Things with Words*. New York: Oxford University Press, 1962. 68

[100] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "KQML as an Agent Communication Language," in *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, N. Adam, B. Bhargava, and Y. Yesha, Eds. Gaithersburg, MD, USA: ACM Press, 1994, pp. 456–463. [Online]. Available: citeseer.ist.psu.edu/article/finin95kqml.html 68

[101] Y. Labrou, "Semantics for an agent communication language," Ph.D. dissertation, Catonsville, MD, USA, 1996, director-Timothy Finin. 68

[102] P. D. O'Brien and R. C. Nicol, "FIPA towards a standard for software agents," *BT Technology Journal*, vol. 16, no. 3, pp. 51–59, 1998. 68

[103] M. Barbuceanu and M. S. Fox, "Cool: A language for describing coordination in multiagent systems," in *Proceedings of the First International Conference oil Multi-Agent Systems (ICMAS-95)*, V. Lesser and L. Gasser, Eds. San Francisco, CA, USA: AAAI Press, 1995, pp. 17–24. [Online]. Available: citeseer.ist.psu.edu/barbuceanu95cool.html 69

[104] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng, "Using colored petri nets for conversation modeling," in *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds. Springer-Verlag: Heidelberg, Germany, 2000, pp. 178–192. [Online]. Available: citeseer.ist.psu.edu/article/cost99using.html 69

[105] M. Alberti, M. Gavanelli, E. Lamma, F. Chesani, P. Mello, and P. Torroni, "A logic based approach to interaction design in open multi-agent systems," Washington, DC, USA, pp. 387–392, Sept. 2004. [Online]. Available: citeseer.ist.psu.edu/alberti04logic.html 69

[106] M.-P. Huget and J. Odell, "Representing agent interaction protocols with agent UML," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 1244–1245. 69

[107] L. Gasser, "An overview of DAI," in *Distributed Artificial Intelligence: Theory and Praxis*, N. M. Avouris and L. Gasser, Eds. Dordrecht: Kluwer, 1992, pp. 9–30. 70

[108] B. Horling, B. Benyo, and V. Lesser, "Using self-diagnosis to adapt organizational structures," in *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA: ACM Press, 2001, pp. 529–536. 70, 76, 81

[109] V. Dignum and F. Dignum, "Modelling agent societies: Co-ordination frameworks and institutions." in *EPIA*, ser. Lecture Notes in Computer Science, P. Brazdil and A. Jorge, Eds., vol. 2258. Springer, 2001, pp. 191–204. 70

[110] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000. 70

[111] J. S. S. Luciano dos Reis Coutinho and O. Boissier, "Modeling organization in MAS: a comparison of models," in *SEAS'05 : 1st. Workshop on Software Engineering for Agent-Oriented Systems*, 2005. 70, 71, 76

[112] M. V. N. Prasad, K. Decker, A. Garvey, and V. Lesser, "Exploring organizational designs with TAEMS: A case study of distributed data processing," in *Proceedings of the First International Conference on Multi–Agent Systems*, V. Lesser, Ed. MIT Press, 1995. [Online]. Available: citeseer.ist.psu.edu/prasad96exploring.html 71

[113] M. Tambe and W. Zhang, "Towards flexible teamwork in persistent teams," in *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*. Washington, DC, USA: IEEE Computer Society, 1998, p. 277. 71

[114] J. S. S. Mahdi Hannoun, Olivier Boissier and C. Sayettat, "MOISE : un modèle organisationnel pour la conception de SMA," in *Proceedings of the 7th Journées Francophones D'Intelligence Artificielle Distribuée et Systèmes Multi-Agents - Ingénierie des Systèmes Multi-Agents Applications*. Reunion Island: Hermès, 1999, pp. 105–118. 71

[115] M. Tambe and W. Zhang, "Towards flexible teamwork in persistent teams: Extended report," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 2, pp. 159–183, 2000. 71

[116] H. V. D. Parunak and J. Odell, "Representing social structures in UML," in *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA: ACM Press, 2001, pp. 100–101. 71

[117] M. Esteva, J. A. Padget, and C. Sierra, "Formalizing a language for institutions and norms," in *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*. London, UK: Springer-Verlag, 2002, pp. 348–366. 71, 74

[118] J. F. Hübner, a. S. Jaime Sim and O. Boissier, "MOISE+: towards a structural, functional, and deontic model for MAS organization," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2002, pp. 501–502. 71

[119] V. T. da Silva, R. Choren, and C. J. P. de Lucena, "A UML based approach for modeling and implementing multi-agent systems," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 914–921. [Online]. Available: http://dx.doi.org/10.1109/AAMAS.2004.36 71

[120] B. Horling and V. Lesser, "A Survey of Multi-Agent Organizational Paradigms," University of Massachusetts, Computer Science Technical Report 04-45, May 2004. [Online]. Available: http://mas.cs.umass.edu/paper/366 71

[121] J. Ferber, F. Michel, and J. Báez Barranco, "Agre : Integrating environments with organizations," in *Environments for Multi-Agent Systems, First International Workshop*. New York, NY, USA: Springer, jul 2004, pp. 48–56. 71, 72

[122] V.-S. J. Dignum V. and D. F., "OMNI: Introducing social structure, norms and ontologies into agent organizations," in *Programming Multi-Agent Systems: Second International Workshop ProMAS 2004*. Berlin Heidelberg: Springer, 2004, pp. 181–198. 71, 73

[123] T. S. José Báez and J. Ferber, "Un modèle institutionnel pour sma organisationnel," in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA05)*. Calais, France: Hermès-Lavoisier, November 2005. 71, 72

[124] J. Ferber and O. Gutknecht, "Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems," in *Proc. of the Third International Conference on Multi-Agent Systems (ICMAS'98)*, 1998. 71, 217

[125] J. F. Hübner, "Um modelo de reorganização de sistemas multiagentes," Ph.D. dissertation, Universidade de São Paulo, Escola Politćnica, Brazil, 2003. 72, 79

[126] J. F. Hübner, J. S. Sichman, and O. Boissier, "Using the MOISE+ for a cooperative framework of MAS reorganisation." in *SBIA*, ser. Lecture Notes in Computer Science, A. L. C. Bazzan and S. Labidi, Eds., vol. 3171. Springer, 2004, pp. 506–515. 72, 76

[127] V. Dignum, J. Vazquez-Salceda, and F. Dignum, "A model of almost everything: Norms, structure and ontologies in agent organizations," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 1498–1499. 73

[128] V. Dignum, J. Vázquez-Salceda, and F. Dignum, "OMNI: Introducing social structure, norms and ontologies into agent organizations," in *PROMAS*, ser. Lecture Notes in Computer Science, R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, Eds., vol. 3346. Springer, 2004, pp. 181–198. 73

[129] V. Dignum, "A model for organizational interaction: based on agents, founded in logic," Ph.D. dissertation, Universiteit Utrecht, 2004. 73

[130] J. Vázquez-Salceda and F. Dignum, "Modelling electronic organizations." in *CEEMAS*, ser. Lecture Notes in Computer Science, V. Marík, J. P. Müller, and M. Pechoucek, Eds., vol. 2691. Springer, 2003, pp. 584–593. [Online]. Available: http://dblp.uni-trier.de/db/conf/ceemas/ceemas2003.html#Vazquez-SalcedaD03 74

[131] P. Mathieu, J. Routier, and Y. Secq, "Principles for dynamic multi-agent organizations," in *5th Pacific Rim International Workshop on Multi Agents*, ser. Lecture Notes in Computer Science, K. Kuwabara and J. Lee, Eds., vol. 2413. Springer, 2002. 77, 83

[132] D. Capera, J.-P. Georgé, M. P. Gleizes, and P. Glize, "The AMAS theory for complex problem solving based on self-organizing cooperative agents," in *WETICE*, 2003, pp. 383–388. 77

[133] G. D. M. Serugendo, "On the use of formal specifications as part of running programs." in *SELMAS*, ser. Lecture Notes in Computer Science, A. F. Garcia, R. Choren, C. J. P. de Lucena, P. Giorgini, T. Holvoet, and A. B. Romanovsky, Eds., vol. 3914. Springer, 2005, pp. 224–237. [Online]. Available: http://dblp.uni-trier.de/db/conf/selmas/selmas2005.html#Serugendo05 78

[134] M. Schillo, H.-J. Bürckert, K. Fischer, and M. Klusch, "Towards a definition of robustness for market-style open multi-agent systems," in *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA: ACM, 2001, pp. 75–76. 78

[135] D. Capera, J.-P. Georgé, M.-P. Gleizes, and P. Glize, "The AMAS theory for complex problem solving based on self-organizing cooperative agents," in *WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies*. Washington, DC, USA: IEEE Computer Society, 2003, p. 383. 78

[136] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996. [Online]. Available: citeseer.ist.psu.edu/dorigo96ant.html 79

[137] S. Brueckner and H. V. D. Parunak, "Self-organizing MANET management." in *Engineering Self-Organising Systems*, ser. Lecture Notes in Computer Science, G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Eds., vol. 2977. Springer, 2003, pp. 20–35. [Online]. Available: http://dblp.uni-trier.de/db/conf/atal/esoa2003.html#BrecknerP03b 79

[138] V. Cahill, E. Gray, J. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen, "Using trust for secure collaboration in uncertain environments," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 52–61, jul 2003. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?2553 79

[139] A. Grizard, L. Vercouter, T. Stratulat, and G. Muller, "A peer-to-peer normative system to achieve social order," in *AAMAS06 Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN'06)*, Hakodate, Japan, May 2006. 79

[140] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003. 79

[141] U. Bellur and N. C. Narendra, "Towards a programming model and middleware architecture for self-configuring systems." in *COMSWARE*. IEEE, 2006. [Online]. Available: http://dblp.uni-trier.de/db/conf/comsware/comsware2006.html#BellurN06 79

[142] J.-B. Welcomme, M.-P. Gleizes, and R. Redon, "A Self-Organising Multi-Agent System Managing Complex System Design Application to Conceptual Aircraft Design," in *International Conference on Complex Open Distributed Systems (CODS), Chengdu, 22/07/2007-24/07/2007*, 2007. 79

[143] V. Chevrier, "Etude et mise en oeuvre du paradigme multi-agents : De ATOME à GTMAS," Ph.D. dissertation, Université Henri Poincaré, Nancy I, 1993. 79

[144] V. Dignum, F. Dignum, and L. Sonenberg, "Towards dynamic reorganization of agent societies," in *Proceedings of the workshop on Coordination in Emergent Agent Societies, Valencia, Spain, August 22-27, 2004*, 2004. 79

[145] Z. Guessoum, "Modèles et architectures d'agents et de systèmes multi-agents adaptatifs," Université Pierre Marie Curie, Paris, December 2003. 80

[146] P. Stone and M. Veloso, "Task decomposition and dynamic role assignment for real-time strategic teamwork," in *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, J. Müller, M. P. Singh, and A. S. Rao, Eds., vol. 1555. Springer-Verlag: Heidelberg, Germany, 1999, pp. 293–308. [Online]. Available: citeseer.ist.psu.edu/article/stone98task.html 82

[147] R. Foisel, V. Chevrier, and J.-P. Haton, "Un modèle pour la réorganisation de systèmes multi-agents," in *Proceedings of the 5ème Journes francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents(JFIADSMA'97)*, 1997. 82

[148] S. A. DeLoach and E. Matson, "An organizational model for designing adaptive multiagent systems," in *Proceedings of the The AAAI-04 Workshop on Agent Organizations: Theory and Practice, San Jose, USA, July 25, 2004*, 2004. 84

[149] E. Matson, "Abstraction of transition properties in multiagent organizations," in *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 169–172. 84

[150] C. M. Jonker, M. C. Schut, and J. Treur, "Organisational change: Deliberation and modification." in *CIA*, ser. Lecture Notes in Computer Science, M. Klusch, S. Ossowski, A. Omicini, and H. Laamanen, Eds., vol. 2782. Springer, 2003, pp. 336–344. 85

[151] B. Horling, "Quantitative Organizational Modeling and Design for Multi-Agent Systems," Ph.D. dissertation, University of Massachusetts at Amherst, February 2006. [Online]. Available: http://mas.cs.umass.edu/paper/409 85

[152] S. Russel and P. Norvig, "Artificial intelligence," 1995. [Online]. Available: citeseer.ist.psu.edu/russel96artificial.html 93

[153] L. Davis, "Bit-climbing, representational bias, and test suite design." in *ICGA*, R. K. Belew and L. B. Booker, Eds. Morgan Kaufmann, 1991, pp. 18–23. 132

[154] S. B. Rana and L. D. Whitley, "Bit representations with a twist." in *ICGA*, T. Bäck, Ed. Morgan Kaufmann, 1997, pp. 188–195. 132

[155] D. Yuret and M. Maza, "Dynamic hillclimbing: Overcoming the limitations of optimization techniques," 1993. [Online]. Available: citeseer.ist.psu.edu/yuret93dynamic.html 132

[156] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986. 132

[157] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with JADE," in *Proc. of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*. Springer-Verlag, 2001, pp. 89–103. 147

[158] D. J. Watts, *Small Worlds – The Dynamics of Networks between Order and Randomness*. Princeton, New Jersey: Princeton University Press, 1999. 180, 181, 183

[159] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Bristol, UK, UK: IOP Publishing Ltd., 1997. 180

[160] P. Ratanchandani and R. Kravets, "A Hybrid Approach to Internet Connectivity for Mobile Ad Hoc Networks," in *Proceedings of IEEE WCNC*, 2003. 180

[161] A. Andronache, M. R. Brust, and S. Rothkugel, "Multimedia Content Distribution in Hybrid Wireless Networks Using Weighted Clustering," in *WMuNeP '06: Proceedings of the 2nd ACM international workshop on Wireless Multimedia Networking and Performance Modeling*. New York, NY, USA: ACM Press, 2006, pp. 1–10. 180

[162] N. I. T. Fujiwara and T. Watanabe, "A Hybrid Wireless Network Enhanced with Multihopping for Emergency Communications," in *ICC '04: Proceedings of the IEEE International Conference on Communications*, 2004, pp. 4177–4181. 180

[163] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in Ad-Hoc and Hybrid Networks," in *INFOCOM*, 2002. 181

[164] A. Helmy, "Small worlds in wireless networks," 2003. [Online]. Available: citeseer.ist.psu.edu/helmy03small.html 181

[165] S. R. K. Alex Reznik and S. Verdu, "A "small world" approach to heterogeneous networks," *Journal of Communications in Informations and Systems (CIS)*, 2003. 181

[166] V. K. Nguyen, "Small-world graphs: Models, analysis and applications in network designs," Ph.D. dissertation, University of California, 2006. 181

[167] G. Sharma and R. Mazumdar, "Hybrid sensor networks: a small world," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2005, pp. 366–377. 181

[168] D. Cavalcanti, D. Agrawal, J. Kelner, and D. F. H. Sadok, "Exploiting the Small-World Effect to Increase Connectivity in Wireless Ad Hoc Networks," in *ICT*, ser. Lecture Notes in Computer Science, J. N. de Souza, P. Dini, and P. Lorenz, Eds., vol. 3124. Springer, 2004, pp. 388–393. 181

[169] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile Computing and Networking*. New York, NY, USA: ACM Press, 2001, pp. 61–69. 181

[170] W. L. D. Lee and J. Kim, "Genetic algorithmic topology control for two-tiered wireless sensor networks," in *Proceedings of the 7th International Conference on Computational Science (ICCS)*. 181

[171] S. Pandey and P. Agrawal, "A unifying architecture for maximal connectivity in heterogeneous ad hoc networks," in *GLOBECOM '06: Proceedings of the IEEE Global Telecommunications Conference*, 2006. 181

[172] K. Herrmann and K. Geihs, "Self-Organization in Mobile Ad hoc Networks based on the Dynamics of Interaction," Erlangen, Germany, 2003, frühjahrstreffen der GI-Fachgruppe Betriebssysteme. [Online]. Available: http://www.kbs.cs.tu-berlin.de/publications/fulltext/gi0403.pdf 183

[173] J. Branke, *Evolutionary Optimization in Dynamic Environments*, ser. Genetic Algorithms and Evolutionary Computation. Kluwer, 2002, vol. 3. 202, 203, 205

[174] K. Weicker, *Evolutionary Algorithms and Dynamic Optimization Problems*. Osnabrück, Germany: Der andere Verlag, 2003. 202

[175] R. W. Morrison, *Designing evolutionary algorithms for dynamic environments*. Springer, 2004. 202

[176] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments: A survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, June 2005. 203

[177] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm." in *PPSN*, ser. Lecture Notes in Computer Science, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., vol. 1498. Springer, 1998, pp. 149–158. 203, 205

[178] R. K. Ursem, "Multinational gas: Multimodal optimization techniques in dynamic environments." in *GECCO*, L. D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, I. C. Parmee, and H.-G. Beyer, Eds. Morgan Kaufmann, 2000, pp. 19–26. 203

[179] E. Alba, J. Saucedo, and G. Luque, "A study of canonical gas for nsops. panmictic versus decentralized genetic algorithms for non-stationary problems," in *Selected Papers from MIC-2005*, ser. Lecture Notes in Computer Science. Springer, 2007. 204

[180] S. Droste, "Analysis of the (1+1) ea for a dynamically changing onemax-variant," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, Eds. IEEE Press, 2002, pp. 55–60. 204

[181] T. Jansen and U. Schellbach, "Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in the lattice," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005, pp. 841–848. 204

[182] D. Arnold and H. Beyer, "Optimum tracking with evolution strategies," *Evolutionary Computing*, vol. 14, no. 3, pp. 291–308, 2006. 204

[183] M. Guntsch and M. Middendorf, "Applying population based aco to dynamic optimization problems," in *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*. London, UK: Springer-Verlag, 2002, pp. 111–122. 204

[184] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem." *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 327–343, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/jco/jco10.html#MontemanniGRD05 204

[185] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *International Conference on Evolutionary Computation*, 1996, pp. 622–627. [Online]. Available: citeseer.ist.psu.edu/gambardella96solving.html 204

[186] X. Q. H. W. L. H. Xiao J., Li J., "Acs-based dynamic optimization for curing of polymeric coating," *AIChE Journal*, vol. 52, no. 4, pp. 1410–1422, 2005. 204

[187] C. M. Fernandes, A. C. Rosa, and V. Ramos, "Binary ant algorithm," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 41–48. 204

[188] X. Hu and R. Eberhart, "Adaptive particle swarm optimisation: detection and response to dynamic systems," in *Proc Congress on Evolutionary Computation*, 2002, pp. 1666–1670. 204

[189] T. Blackwell and P. Bentley, "Don't push me! collision-avoiding swarms," *cec*, vol. 02, pp. 1691–1696, 2002. 204

[190] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems." in *EvoWorkshops*, ser. Lecture Notes in Computer Science, G. R. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, Y. Jin, C. G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, and G. Squillero, Eds., vol. 3005. Springer, 2004, pp. 513–524. [Online]. Available: http://dblp.uni-trier.de/db/conf/evoW/evoW2004.html#JansonM04 204

[191] X. Li and K. Dam, "Comparing particle swarms for tracking extrema in dynamic environments," in *Congress on Evolutionary Computation*. IEEE, 2003, pp. 1772–1779. 204

[192] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 20-23 June 2004, pp. 98–103. 204

[193] J. Branke and T. Blackwell, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, ser. LNCS, vol. 3005. Springer, 2004, pp. 489–500. 204

[194] K. E. Parsopoulos and M. N. Vrahatis, "Unified particle swarm optimization in dynamic environments," in *EvoWorkshops*, ser. Lecture Notes in Computer Science, F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith, and G. Squillero, Eds., vol. 3449. Springer, 2005, pp. 590–599. 204

[195] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *Congress on Evolutionary Computation*. Singapore: IEEE, 25-28 September 2007, pp. 564–567. 204

[196] R. Mendes and A. Mohais, "Dynde: Differential evolution for dynamic optimization problems," in *Congress on Evolutionary Computation*. Edinburgh, UK: IEEE Computer Society, September 2005, pp. 583–590. 204

[197] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of Congress on Evolutionary Computation CEC99*. IEEE, 1999, pp. 1875–1882. 205

[198] K. T. Handa Hisashi, Katai Osamu and B. Mitsuru, "Solving dynamic csps by coevolutionary ga," *Chino Shisutemu Shinpojiumu Shiryo*, vol. 26, pp. 63–68, 1999. 205

[199] E. Hart and P. Ross, "An immune system approach to scheduling in changing environments," in *GECCO*, W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 1559–1566. 205

[200] R. W. Morrison, "Performance measurement in dynamic environments," in *GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, A. M. Barry, Ed., July 2003, pp. 99–102. 206

[201] K. Weicker, "Performance measures for dynamic environments," in *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 2002, pp. 64–76. 206

[202] "Internet repository on evolutionary algorithms for dynamic optimizaion problems." [Online]. Available: http://www.aifb.uni-karlsruhe.de/~jbr/EvoDOP/references.html 206

[203] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Trans. Mobile Computing*, vol. 2, no. 3, pp. 257–269, July–September 2003. [Online]. Available: citeseer.ist.psu.edu/bettstetter03node.html 208

[204] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. 211, 215

[205] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, Tech. Rep. 103, 2001. 211, 215

[206] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "A cellular genetic algorithm for multiobjective optimization," in *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, 2006, pp. 25 – 36. 211, 215

[207] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multiobjective evolutionary algorithm," in *MICAI*, ser. Lecture Notes in Computer Science, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and J. H. S. Azuela, Eds., vol. 2972. Springer, 2004, pp. 688–697. 215

[208] S. A. Deloach, "Analysis and design using MaSE and AgentTool," in *Proc. of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001)*, Miami University, Oxford, Ohio, 2001. 217

[209] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000. 217

[210] J. C. Collis, D. T. Ndumu, H. S. Nwana, and L. C. Lee, "The ZEUS agent building tool-kit," *BT Technology Journal*, vol. 16, no. 3, pp. 60–68, 1998. 217

[211] S. DeLoach and M. F. Wood, "Developing multiagent systems with AgentTool," in *ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*. London, UK: Springer-Verlag, 2001, pp. 46–60. 217

[212] E. Norling and F. E. Ritter, "Embodying the JACK agent architecture," in *AI '01: Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*. London, UK: Springer-Verlag, 2001, pp. 368–377. 217

[213] O. Gutknecht and J. Ferber, "The Madkit agent platform architecture," in *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems*. London, UK: Springer-Verlag, 2001, pp. 48–55. 217

[214] P.-M. Ricordel and Y. Demazeau, "From analysis to deployment: A multi-agent platform survey," in *ESAW '00: Proceedings of the First International Workshop on Engineering Societies in the Agent World*. London, UK: Springer-Verlag, 2000, pp. 93–105. 217

[215] T. G. Nguyen and T. T. Dang, "Agent platform evaluation and comparison," Institute of Informatics, Slovak Academy of Sciences, Tech. Rep., june 2002. 217

[216] R. Leszczyna, "Evaluation of agent platforms," European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Tech. Rep., june 2004. 217

[217] G. Danoy, "An agent framework for optimization using coevolutionary genetic algorithms," Master's thesis, Ecole Nationale Superieure des Mines de Saint-Etienne (ENMSE), 2004. 222

[218] Y. Jun and E. Shakshuki, "Performance evaluation of agent toolkits." in *Canadian Conference on AI*, ser. Lecture Notes in Computer Science, A. Y. Tawfik and S. D. Goodwin, Eds., vol. 3060. Springer, 2004, pp. 556–558. [Online]. Available: http://dblp.uni-trier.de/db/conf/ai/ai2004.html#JunS04 222

[219] D. Burbeck, K. Garpe and S. Nadjm-Tehrani, "Scale-up and performance studies of three agent platforms," in *IPCCC '04: Proceedings of the International Performance, Computing, and Communications Conference*, 2004, pp. 857–863. 222

[220] W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, Eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), 13-17 July 1999, Orlando, Florida, USA*. Morgan Kaufmann, 1999.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 482 I

Grégoire Danoy

Title: A Multi-Agent Approach for Hybrid and Dynamic Coevolutionary Genetic Algorithms : Organizational Model and Real-World Problems Applications

Speciality : Computer Science

Keywords : Multi-Agent System, Optimization, Genetic Algorithms, Real-World Applications

Abstract :

Since the mid 1970s and the introduction of Genetic Algorithms (GAs) by John H. Holland, the idea of mimicking the capacity of biological systems to adapt to the genetic level in response to environmental challenges has motivated many research studies for applying similar mechanisms to scientific problems. One recent evolution of such algorithms, namely Coevolutionary Genetic Algorithms (CGAs), focuses on the coevolution of populations (competing or cooperating) of individuals representing specific parts of the global solution instead of evolving a population of similar individuals representing a global solution. This thesis work aims at modeling and applying such CGAs as well as developing new ones in the context of business problems optimization.

In this dissertation we assert that modeling CGAs as organizational multi-agent systems overcomes the lack of explicitness at the level of the algorithms structure, interactions and adaptation to existing models and platforms. We therefore introduce MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, a new agent organizational and reorganizational model based on Moise+ and dedicated to evolutionary optimization. This model was used to describe existing CGAs as well as to build two new variants, hybrid and dynamic, of a competitive CGA.

This thesis also presents the design and implementation of DAFO, a Distributed Agent Framework for Optimization, permitting the use, the manipulation and the distribution of CGAs. Modeled using MAS4EVO and built on top of a multi-agent platform, DAFO allows the application and comparison of various CGAs (existing and novel ones) on optimization problems.

The CGAs experimentations were conducted on two business problems. The first one is an existing inventory management problem for which we studied multiple static instances. We demonstrated the added value of decomposition on small problem instances as well as the improvement brought by the new hybrid and dynamic CGAs. The second problem studied is a new topology control problem in wireless ad hoc networks for which a first mathematical model has been created. State-of-the-art results were obtained while evaluating the performance of different CGAs on multiple static instances and on one dynamic instance of this network optimization problem.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 482 I

Grégoire Danoy

Titre de la thèse : Une approche multi agent pour les algorithmes génétiques coévolutionnaires hybrides et dynamiques: modèle d'organisation multi-agent et mise en œuvre sur des problèmes métiers

Spécialité : Informatique

Mots clefs : Système Multi-Agent, Optimisation, Algorithmes Génétiques, Problèmes métier

Résumé :

Depuis le début des années 1970 et l'introduction des Algorithmes Génétiques (AG) par John H. Holland, l'idée de mimer la capacité d'adaptation au niveau génétique des systèmes biologiques en réponse à des modifications environnementales a motivé de nombreuses recherches utilisant des mécanismes similaires pour des problèmes scientifiques. Une évolution récente de tels algorithmes, appelés Algorithmes Génétiques Coévolutionnaires (AGCs), s'intéresse à la coévolution de populations d'individus (en coopération ou en compétition) représentants des parties spécifiques de la solution globale au lieu d'évoluer une population d'individus similaires représentants la solution globale. Cette thèse a pour objectif de modéliser et d'appliquer de tels AGCs ainsi que d'en développer de nouveaux dans le contexte de l'optimisation de problèmes métier.

Nous défendons la thèse selon laquelle la modélisation des AGCs sous forme de systèmes multi-agent organisationnels répond au manque d'expressivité en terme de structure, d'interactions et d'adaptation de ces algorithmes dans les modèles et plateformes existants. Dans cette optique nous introduisons MAS4EVO, Multi-Agent Systems for EVolutionary Optimization, un nouveau modèle agent organisationnel et réorganisationnel basé sur Moise+ et dédié à l'optimisation évolutionnaire. Ce modèle a été utilisé pour décrire et mettre en oeuvre de tels AGCs ainsi que pour construire deux nouvelles variantes, hybride et dynamique, d'un AGC compétitif.

Cette thèse présente également la modélisation et l'implémentation de DAFO (Distributed Agent Framework for Optimization), un framework multi-agent organisationnel permettant l'utilisation, la manipulation et la distribution d'AGCs. Modélisé à l'aide de MAS4EVO et construit sur base d'une plateforme agent existante, il permet d'appliquer et de comparer différents AGCs (existants et nouveaux) sur des problèmes d'optimisation difficiles.

Les expérimentations de ces AGCs ont été conduites sur deux problèmes d'optimisation métier. Le premier est un problème de gestion de stock pour lequel différentes instances statiques ont été étudiées. Nous avons démontré l'apport de la décomposition sur des instances de petite taille ainsi que l'amélioration procurée par les nouveaux AGCs hybrides et dynamiques. Le second problème étudié est un problème de contrôle de topologie dans les réseaux ad hoc sans fil pour lequel une première modélisation mathématique a été réalisée. Des résultats de pointe ont été obtenus lors de l'évaluation des performances de différents AGCs sur de multiples instances statiques et sur une instance dynamique de ce problème d'optimisation de réseaux.