Techniques d'optimisation déterministe et stochastique pour la résolution de problèmes difficiles en cryptologie

Sarra Bouallagui

Thèse

en vue de l'obtention du titre de

DOCTEUR DE L'INSTITUT NATIONAL
DES SCIENCES APPLIQUEES DE ROUEN

(arrêté ministériel du 7 août 2006)

présentée par

Sarra BOUALLAGUI

Titre de la thèse :

Techniques d'optimisation déterministe et stochastique pour la résolution de problèmes difficiles en cryptologie

Date de la soutenance: 05/07/10
Composition du jury :

Président		
Adnan	YASSINE	Professeur, Université du Havre
Rapporteurs		
Ibrahima	SAKHO	Professeur, Université de Metz
Jin Kao	HAO	Professeur, Université d'Angers
Examinateurs		
Mohamed	DIDI BIHA	Professeur, Université de Caen
Hoai An	LE THI	Professeur, Université de Metz (Directrice de Thèse)
Traian	MUNTEAN	Professeur, Université de la Méditerranée-Marseille
Hung	NGUYEN VIET	Maitre des Conférences, Université Paris 6
Tao	PHAM DINH	Professeur, INSA de Rouen (Directeur de thèse)

En souvenir de ma mère :
Maman, une femme aussi adorable que toi je n'en connais pas, tu as toujours été là pour moi, et à aucun moment tu n'as cessé de me couvrir de ta tendresse.

Remerciements

Je tiens à remercier

Madame LE Thi Hoai An, ma Directrice de thèse, pour l'aide compétente qu'elle m'a apportée, pour sa patience, son soutien permanent, ses conseils précieux, ses encouragements et pour sa disponibilité pour répondre à mes questions. Son oeil critique m'a été très précieux pour structurer mon travail.

Monsieur Pham Dinh Tao, le fondateur de la programmation DC et DCA et mon co-directeur de thèse, pour ses conseils.

Les membres de jury pour avoir accepté d'évaluer mon travail.

Je n'oublie pas mes collègues de LMI où, pendant plus de quatre ans, j'ai pu profiter de leurs expériences et de leurs grandes compétences.

Je ne pourrai jamais oublier le soutien et l'aide des personnes chères de ma merveilleuse famille. Je réserve une reconnaissance particulière à ma mère avec la grande douleur de ne plus l'avoir parmi nous, pour le soin incomparable dont elle a fait preuve malgré son mal.

Un grand merci à mon mari, à mon fils et à tous mes amis pour leurs encouragements, amour et affection.

Enfin à tous ceux qui m'ont soutenu de près ou de loin et à tous ceux qui m'ont incité même involontairement à faire mieux, veuillez trouver ici le témoignage de ma profonde gratitude.

Sarra Bouallagui
4, rue Charles Péguy
77500, Chelles

Tél. : 06 69 34 43 48
E-mail : sarra.bouallagui@gmail.com

Doctorante en Optimisation et Recherche Opérationnelle

Formation

2006-2010 : Thèse de Doctorat en informatique (INSA de Rouen)

2005-2006 : Master Modélisation par les logiciels (Université Paul Verlaine-Metz)

2001-2004 : Diplôme d'ingénieur en informatique (Faculté de sciences de Tunis)

1993-1999 : Baccalauréat Scientifique (Lycée Pilote de Gafsa)

Expérience professionnelle

Depuis 09/2006 Laboratoire de mathématique et informatique (LMI) de l'INSA de Rouen :
Doctorante en recherche opérationnelle et techniques d'optimisation : techniques d'optimisation pour la sécurité informatique

- Etude bibliographique des approches existantes pour les algorithmes de cryptage basés sur des problèmes mathématiques

- Evaluation de la sûreté et la fiabilité en conduisant des tentatives d'attaque utilisant les techniques d'optimisation : étude d'algorithmes spécifiques, proposition d'une solution, conception et développement dans l'environnement c++ ; Ilog Cplex ; tests unitaires et tests fonctionnels

De 02/2006 à 07/2006 Laboratoire d'Informatique Théorique et appliquée (LITA) université Paul Verlaine de Metz :
Stage de Master 2 Modélisation par les logiciels : construction de fonctions booléennes équilibrées de haut degré de non linéarité en cryptographie

- Etat de l'art des fonctions booléennes cryptographiques
- Conception et développement d'un algorithme génétique pour la construction des fonctions booléennes cryptographiquement fortes dans l'environnement c++, Ilog Cplex, tests unitaires et tests fonctionnels

De 09/2004 à 09/2005 IMITECH, Tunisie
Développeur *Ccharp.net* sur un projet d'implémentation d'un comparateur de prix www.acheter-pas-cher.com

- Conception et développement dans l'environnement *MySQL/Ccharp.net*
- Tests unitaires, tests d'intégration ; corrections d'anomalies

Développeur *Ccharp.net* sur un projet d'implémentation d'une application pour la mesure de retour sur investissement en utilisant la technologie des marqueurs

- Analyse de besoins MOA et propositions de solutions
- Rédaction de spécifications fonctionnelles détaillées
- Développements dans l'environnement *MySQL/Ccharp.net*
- Tests unitaires ; tests d'intégration ; corrections d'anomalies

De 02/2004 à 06/2004 Medianet, Tunisie
Projet de fin d'études : contribution au développement d'une application pour la mesure d'audience

- Analyse du besoin et proposition de solution
- Rédaction de spécifications fonctionnelles
- Développement dans l'environnement JSP/Oracle 8i
- Elaboration des scénarios de tests
- Tests unitaires ; tests d'intégration ; correction d'anomalies

De 07/2003 à 08/2003 Tunisie Télécom, Tunisie
Stage d'été : mise en place d'une politique de sécurité pour un réseau local

- Etude d'un réseau existant et les failles qu'il présente
- Mise en place d'une politique de sécurité en utilisant LANguard sous windows 2000

Compétences techniques

Langage de programmation : C/C++, Java, JSP, Java Script, Visual Basic, Pascal, PL/SQL, Prolog, Caml, Ccharp, HTML, Asp.net..

Systèmes d'exploitation : Windows 9x/ XP/2000, Linux

Bases de données : Access, Oracle, MySql

Méthodes de Conception : Merise, UML

Logiciels de programmation linéaire : Ilog Cplex - Xpress

La liste de publications

1. Sarra Bouallagui, Hoai An Le Thi and Tao Pham Dinh, Cryptanalysis of an Identification Scheme Based on the Perceptron Problem using a hybridization of deterministic optimization and genetic algorithm ; Proceedings of the 2009 International Conference on Security and Management ; pp. 117-123, World Congress in Computer Science Computer Engineering, and Applied Computing ; July 13-16, 2009 ; Las Vegas, USA
2. Sarra Bouallagui, Hoai An Le Thi and Tao Pham Dinh, Design of highly nonlinear balanced Boolean functions using an hybridation of DCA and Simulated Annealing algorithm in Modeling, Computation and Optimization in Information Systems and Management Sciences, Communications in Computer and Information Science CCIS Volume 14 Springer, pp. 583-592

3. Sarra Bouallagui, Hoai An Le Thi and Pham Dinh Tao : Design of bent boolean functions using an Hybridization of DCA and Simulated Annealing Algorithm, International Conference on Computational Management Science ; Vienna, Austria ; July 28-30, 2010 (abstract)

Langues

Arabe : Langue maternelle
Anglais : TOEIC 750pts (2009)
Italien : Scolaire

Le résumé

Cette thèse s'articule autour des fonctions booléennes liées à la cryptographie et la cryptanalyse de certains schémas d'identification. Les fonctions booléennes possèdent des propriétés algébriques fréquemment utilisées en cryptographie pour constituer des S-Boxes (tables de substitution).
Nous nous intéressons, en particulier, à la construction de deux types de fonctions : les fonctions courbes et les fonctions équilibrées de haut degré de non-linéarité.
Concernant la cryptanalyse, nous nous focalisons sur les techniques d'identification basées sur les problèmes de perceptron et de perceptron permuté. Nous réalisons une nouvelle attaque sur le schéma afin de décider de sa fiabilité.
Nous développons ici des nouvelles méthodes combinant l'approche déterministe DCA (Difference of Convex functions Algorithm) et heuristique (recuit simulé, entropie croisée, algorithmes génétiques...). Cette approche hybride, utilisée dans toute cette thèse, est motivée par les résultats intéressants de la programmation DC.

Mots clé : Programmation DC et DCA, recuit simulé, les algorithmes génétiques, la méthode d'entropie croisée, hybridation, fonctions booléennes, équilibre, fonctions courbes, non-linéarité, schéma d'identification, problème perceptron, programmation DC polyédrale en 0-1...

Abstract

In cryptography especially in block cipher design, boolean functions are the basic elements. A cryptographic function should have high non-linearity as it can be attacked by linear method.

There are three goals for the research presented in this thesis :
- Finding a new construction algorithm for the highest possible nonlinear boolean functions in the even dimension, that is bent functions, based on a detreministic model.
- Finding highly non linear boolean functions.
- Cryptanalysing an identification scheme based on the perceptron problem.

Optimisation heuristic algorithms (Genetic algorithm and simulated annealing) and a deterministic one based on DC programming (DCA) were used together.

KeyWords : *boolean function, non linearity, balance, genetic algorithm, cross entropy method, hybridization, simulated annealing, DCA, DC programming, polyhedral 0-1 DC programming, bent functions, perceptron problem, identification scheme... ...*

Table des matières

Table des figures

Liste des tableaux

Introduction

« Le désir humain de perfection trouve son expression dans la théorie de l'optimisation. Elle étudie comment décrire et atteindre ce qui est meilleur, une fois que l'on connaît comment mesurer et modifier ce qui est bon et ce qui est mauvais . . . la théorie de l'optimisation comprend l'étude quantitative des optimums et les méthodes pour les trouver»[2].

A partir de la citation ci-dessus, on comprend la motivation d'un processus d'optimisation. En fait, l'optimisation cherche à améliorer une performance en se rapprochant d'un point optimum.
Selon la nature de la fonction objectif et de l'ensemble contraint, on distingue deux classes de problèmes : les problèmes convexes dont on dispose aujourd'hui d'un arsenal théorique et numérique considérable pour les résoudre, et les problèmes non convexes dont l'étude est en plein essor.

L'optimisation non convexe connaît, au cours des deux dernières décennies, des développements spectaculaires. Il y a une raison simple et évidente : la plupart des problèmes d'optimisation de la vie courante sont de nature non convexe. On peut distinguer deux approches différentes mais complémentaires en programmation non-convexe :

i. Approches globales combinatoires : qui sont basées sur les techniques combinatoires de la Recherche Opérationnelle. Elles consistent à localiser les solutions optimales à l'aide des méthodes d'approximation, des techniques de coupe, des méthodes de décomposition, de séparation et évaluation. Ces approches ont connu un développement important au cours de ces dernières années. L'inconvénient majeur des méthodes globales est leur lourdeur et leur coût trop important. Elles ne peuvent pas être appliquées aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.

ii. Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. La programmation DC (Différence de deux fonctions Convexes) et son algorithme DCA (DC Algorithme) sont des techniques très importantes de ces approches.
Elles ont été introduites par T.Pham Dinh en 1985 à l'état préliminaire et développés intensivement à travers de nombreux travaux communs de H.A Le Thi et T. Pham Dinh depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens dans différents domaines des sciences appliquées (voir [3], [4], [5], [6], [7], [6], [8],

[9], [10], [11], [12] et les références incluses).

Un programme DC est de la forme

$$\alpha = \inf \{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \, (P_{dc})$$

où $g, h \in \Gamma_0(\mathbb{R}^n)$, le cône convexe de toutes les fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. Une telle fonction f est appelée fonction DC et g et h des composantes DC de f. La programmation DC est une extension de la Programmation Convexe : cette extension est assez large pour couvrir la quasi-totalité des programmes non convexes dits réalistes mais pas trop pour pouvoir utiliser l'arsenal puissant de la Programmation Convexe. DCA est une approche locale qui travaille avec les deux fonctions convexes (dont la différence est la fonction objectif elle-même du programme DC) et non avec cette fonction objectif. Puisqu'une fonction DC admet une infinité de décompositions DC, il y a une infinité de DCA appliqués à un programme DC. Les impacts de ces décompositions DC sur les qualités des DCA correspondants (rapidité, robustesse, globalité, ...) sont importants. La résolution d'un problème concret par DCA devrait répondre aux deux questions cruciales :

- La recherche d'une bonne décomposition DC : cette question est largement ouverte. Les techniques de reformulation sont souvent utilisées et très efficaces pour l'obtention des décompositions DC intéressantes.
- La recherche d'un bon point initial : cette recherche est basée sur la combinaison de DCA avec les méthodes globales de type Séparation et Evaluation (SE) et/ou Approximation de l'Extérieur (AE), sur l'hybridation de DCA et les algorithmes heuristiques.

Cette thèse est consacrée aux techniques d'optimisation non convexe basées sur l'hybridation de l'algorithme dédié à *la programmation DC* avec des approches heuristiques pour certains problèmes de *cryptologie*. Cette hybridation a pour but de remédier aux inconvénients des techniques heuristiques.
La cryptographie est l'étude des systèmes mathématiques propres à résoudre les problèmes de sécurité tels que la confidentialité et l'authentification. Actuellement, la définition la plus répandue de la cryptographie concerne la communication en présence d'attaquants.
A côté du terme ancien de cryptographie, limité à l'origine au chiffrement des messages, on a introduit récemment, de manière à préciser la terminologie et rendre compte de l'évolution récente du domaine, la dénomination générale de cryptologie pour désigner la partie de la sécurité des systèmes d'information qui s'occupe d'assurer, ou au contraire de compromettre, les grandes fonctions de sécurité.
Ainsi la cryptologie a un double aspect : d'une part, la cryptographie qui consiste à construire des outils propres à assurer des fonctionnalités de sécurité ; d'autre part la cryptanalyse qui consiste à s'attaquer aux outils en question pour en trouver les faiblesses.

Nous nous sommes intéressés dans cette thèse aux deux aspects de la cryptologie.

La cryptographie symétrique, également dite à clé secrète (par opposition à la cryptographie à clé publique), est la plus ancienne forme de chiffrement.

On peut citer DES (Data Encyption Standard) comme étant l'un des algorithmes les plus connus qui se basent sur le chiffrement par blocs qui est une des deux grandes catégories de chiffrements modernes en cryptographie symétrique.
Pour assurer la transformation de texte en clair à un texte chiffré ou cryptogramme, DES utilise les S-boxes (boîtes de substitution). La fonctionnalité de base des S-boxes est de prendre en entrée un nombre m de bits et les transformer en un autre nombre n de bits afin d'avoir une transformation qui ne peut pas être facilement trouvée par les cryptanalyses. Les fonctions booléennes assurant cette fonctionnalité doivent avoir de bonnes propriétés cryptographiques : une valeur importante de non linéarité et déséquilibre nul (fonction équilibrée). Les propriétés de la fonction booléenne courbe lui donnent une certaine résistance à la cryptanalyse différentielle et à la cryptanalyse linéaire. Par conséquent, ces fonctions semblent être, alors, le meilleur choix pour la construction de S-Boxs.

La recherche de telles fonctions booléennes ne cesse d'occuper les chercheurs, d'ailleurs plusieurs approches ont été fondées en utilisant les différentes techniques d'optimisation : méta heuristique, déterministe...

C'est dans ce cadre que s'inscrit la première partie de notre travail. Nous contribuons au développement de deux algorithmes destinés à la génération de fonctions booléennes courbes et les fonctions booléennes équilibrées de haut degré de non-linéarité.

Le problème de fonctions courbes est modélisé comme un problème quadratique concave à contraintes linéaires. Notre travail consiste à faire :

1- La modélisation DC de problème, formulations de programme DC, le choix des décompositions DC les mieux adaptées.

2- Les implémentations et simulations numériques pour leurs validations.

Pour les fonctions booléennes de haut degré de non-linéarité, nous allons utiliser un modèle d'optimisation déterministe présenté dans [1] qui consiste à la minimisation d'une fonction convexe polyédrale sur un polyèdre convexe en variables 0-1. Grâce à la pénalité exacte, ce problème a été reformulé en programmation DC.
Notre contribution consiste à :

- étudier plusieurs versions de combinaison des approches DCA et le recuit simulé (SA) dans le but d'exploiter simultanément l'efficacité de chaque approche.
- chercher une bonne fonction de départ pour l'algorithme SA-DCA en utilisant une fonction courbe.
- étudier la combinaison DCA et entropie croisée.

La deuxième partie est consacrée à la cryptanalyse à travers l'étude cryptanalytique d'un schéma d'identification basé sur les deux problèmes "Perceptron" et "Perceptron Permuté" (en Anglais : "Perceptron Problem" (PP) et "Permuted Perceptron Problem" (PPP)). PPP est un schéma d'authentification proposé en 1995 par David Pointcheval [13] en se basant sur le problème NP-complet PP qui doit son existence au fameux problème perceptron dans les réseaux de neurones. Plusieurs tentatives d'attaques ont été réalisées sur les problèmes PP et PPP pour évaluer la sécurité des protocoles d'identification basés sur PPP. Certaines attaques basées sur le recuit simulé ont été développées depuis 1999. En 2007, un nouveau modèle d'optimisation déterministe a été développé pour chacun de ces deux problèmes [1]. PP a été formulé comme la minimisation d'une fonction DC polyédrale sur un polyèdre convexe. PPP a été modélisé de manière analogue à PP et résolu par DCA.
Notre contribution porte sur la résolution numérique de problèmes : nous avons utilisé plusieurs versions de combinaisons entre GA, DCA et SA.

Il est à noter que tous les quatre problèmes étudiés sont des problèmes combinatoires NP-difficiles et de très grande dimension en pratique. Nos contributions propres portent sur les méthodes de résolution.

Organisation de la thèse

Cette thèse est divisée en deux parties et est composée de neuf chapitres.
Dans la première partie intitulée "méthodologie" nous présentons des outils théoriques et algorithmiques servant de références pour la suite et nous étudions en profondeur la programmation DC et DCA qui seront utilisés tout au long de cette thèse. Le chapitre 5 est dédié aux techniques d'hybridation que nous avons utilisées dans nos travaux.

La deuxième partie commence par le chapitre 6, qui est une introduction à la cryptologie. Dans le chapitre 7, nous présentons les fonctions courbes et nous décrivons un nouvel algorithme hybride basé sur DCA dédié à la génération de ces fonctions.
Dans le chapitre 8, nous détaillons la génération des fonctions booléennes équilibrées de haut degré de non-linéarité par l'algorithme combinant le recuit simulé et DCA.

Le chapitre 9 décrit la cryptanalyse d'un schéma d'identification basé sur les deux problèmes PP et PPP en combinant DCA à des techniques heuristiques.
Tous les chapitres de la deuxième partie se terminent par les résultats des tests numériques.

Première partie

Méthodologies

Chapitre 1

Programmation DC

1.1 Eléments d'analyse convexe et conditions d'optimalité

Dans cette section nous rappellerons brièvement quelques notions d'analyse convexe qui nous seront utiles pour la suite de notre exposé. Nous ne chercherons pas à être exhaustif. Des développements sur le sujet peuvent être trouvés dans la monographie de Rockafellar [14], et l'ouvrage de Hiriart-Urruty et Lemaréchal [15] dans lesquels la théorie est détaillée dans le cadre de la dimension finie. Le livre d'Ekeland et Temam [16] est quant à lui consacré au cadre de la dimension infinie.
Dans la suite, $\mathcal{X}$ désignera l'espace euclidien $\mathbb{R}^n$, muni du produit scalaire usuel noté $\langle ., . \rangle$et de la norme euclidienne associée $\|x\| = \sqrt{\langle x, x \rangle}$. Et $\mathcal{Y}$ désignera l'espace vectoriel dual de $\mathcal{X}$ relatif au produit scalaire, que l'on pourra identifier à $\mathcal{X}$. On notera également

$$\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\},$$

muni d'une structure algébrique déduite de celle de $\mathbb{R}$ avec la convention

$$+\infty - (+\infty) = +\infty. \tag{1.1}$$

1.1.1 Ensembles convexes

Pour une partie $\mathcal{C}$ de $\mathcal{X}$, on dit que $\mathcal{C}$ est convexe lorsque, chaque fois qu'on y prend $x, y \in \mathcal{C}$ le segment $[x, y]$ tel que

$$[x, y] = \{(1 - t)\, x + ty | t \in [0, 1]\},$$

qui les joint y est entièrement contenu. Soit $S \subset \mathcal{X}$, l'enveloppe convexe de S notée $co(S)$, est l'ensemble des combinaisons convexes finies d'éléments de S, c'est à dire

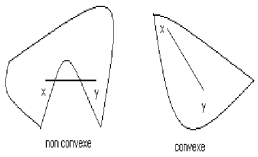


FIGURE 1.1 – Un ensemble convexe

$$co(S) = \left\{ \sum_{i=0}^{n} \lambda_i x^i | \lambda_i \in \mathbb{R}^+, x^i \in S \ et \ \sum_{i=0}^{n} \lambda_i = 1 \right\}.$$

Soit $\mathcal{C}$ un ensemble convexe de $\mathbb{R}^n$, on définit la variété linéaire engendrée par $\mathcal{C}$ comme étant l'ensemble

$$aff(\mathcal{C}) = \left\{ \sum_{i=0}^{n} \lambda_i x^i | \lambda_i \in \mathbb{R}, x^i \in \mathcal{C} \ et \ \sum_{i=0}^{n} \lambda_i = 1 \right\}.$$

On montre que $aff(\mathcal{C})$ est le translaté d'un espace vectoriel. On appelle *intérieur relatif* d'un ensemble convexe $\mathcal{C}$, son intérieur dans $aff(\mathcal{C})$, muni de la topologie induite de celle de $\mathcal{X}$. On le note

$$ir(\mathcal{C}) = \{x \in \mathcal{C} | \ \exists r \geq 0 \ tel \ que \ B(x,r) \cap aff(\mathcal{C}) \subset \mathcal{C}\},$$

où $B(x,r)$ est la boule [1] centrée en x et de rayon r.

Remarque 1.1.1
$ir(\mathcal{C})$ est vide si et seulement si $\mathcal{C}$ l'est.

1.1.2 Fonctions convexes

Soit $f : \mathcal{C} \longrightarrow \overline{\mathbb{R}}$ une fonction définie sur un ensemble convexe $\mathcal{C}$ de $\mathcal{X}$; on appelle domaine de f l'ensemble

$$dom(f) = \{x \in \mathcal{C} | f(x) < +\infty\}.$$

La fonction f est dite *propre* si elle ne prend jamais la valeur $-\infty$ et si elle n'est pas identiquement égale à $+\infty$.
L'épigraphe de f est l'ensemble

$$epi(f) = \{(x,\alpha) \in \mathcal{C} \times \mathbb{R} | f(x) \leq \alpha\}.$$

1. $B(x,r) = \{y : \|x - y\| \leq r\}$

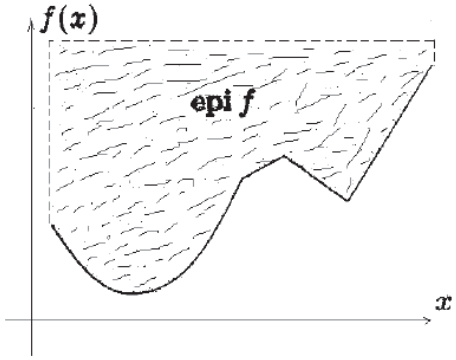


FIGURE 1.2 – Epigraphe d'une fonction.

Un exemple d'épigraphe d'une fonction est représenté sur la figure 1.2.
La fonction f est dite *convexe* si son épigraphe est un ensemble convexe de $\mathbb{R} \times \mathcal{X}$. Ceci équivaut à dire que l'on à l'inégalité

$$f\left((1-\lambda)x + \lambda y\right) \leq (1-\lambda)f(x) + \lambda f(y)$$
$$\text{pour tout } x, y \in \mathcal{C} \text{ et tout } \lambda \in [0,1]\,. \quad (1.2)$$

Si l'inégalité est stricte dans (1.2) pour tout $\lambda \in]0;1[$ et pour tout $x, y \in \mathcal{C}$ avec $x \neq y$ alors la fonction f est dite *strictement convexe*. On dira que la fonction f est *fortement convexe* de module ρ sur l'ensemble convexe $\mathcal{C}$ s'il existe un nombre réel $\rho \geq 0$ tel que

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) - (1-\lambda)\lambda\frac{\rho}{2}\|x-y\|^2$$
$$\text{pour tout } x, y \ \in \ \mathcal{C} \text{ et tout } \lambda \in [0,1]\,. \quad (1.3)$$

Plus précisément, f est fortement convexe sur le convexe $\mathcal{C}$ si

$$\rho(f, \mathcal{C}) = \sup\left\{\rho \geq 0 | f - \frac{\rho}{2}\|.\| \text{ est convexe sur } \mathcal{C}\right\}. \quad (1.4)$$

Notons que si $\rho(f, \mathcal{C}) > 0$ alors (1.2) est vérifiée pour tout $\rho \in [0, \rho(f, \mathcal{C})[$.
On dira que la borne supérieure est atteinte dans la définition (1.4) si $f - \frac{\rho}{2}$ est convexe sur $\mathcal{C}$ tout entier. Si $\mathcal{C} = \mathcal{X}$, alors on notera $\rho(f)$ au lieu de $\rho(f, \mathcal{C})$.

Remarque 1.1.2
Toute fonction fortement convexe est strictement convexe et toute fonction strictement convexe est convexe.

Etant donné une fonction convexe propre f sur $\mathcal{X}$, on dira que $y^0 \in \mathcal{Y}$ est un *sous gradient* de f en $x^0 \in dom(f) si$

$\langle y^0, x - x^0 \rangle + f(x^0) \leq f(x) \forall x \in \mathcal{X}.$

L'ensemble de tous les sous-gradients de f au point x^0 est le *sous-différentiel* de f au point x^0 noté $\partial f(x^0)$. Le domaine du sous-différentiel de la fonction f est

$$dom(\partial f) = \{x | \partial f(x) \neq \oslash\}. \tag{1.5}$$

Soit $\mathcal{E}$ un réel positif, un élément y^0 de $\mathcal{Y}$ est appelé $\mathcal{E}$-sous-différentiel de f au point x^0 si

$$\langle y^0, x - x^0 \rangle + f(x^0) - \mathcal{E} \leq f(x) \forall x \in \mathcal{X}.$$

On note $\partial_{\mathcal{E}} f(x^o)$ l'ensemble de tous les $\mathcal{E}$-sous-différentiels de f au point x^o. La fonction f est dite *semi-continue inférieurement*(s.c.i) en un point $x \in \mathcal{C}$ si

$$\liminf_{y \to x} f(y) \geq f(x).$$

On note $\Gamma_0(\mathcal{X})$ l'ensemble des fonctions convexes s.c.i et propres sur $\mathcal{X}$. On définit la fonction f^*, *conjuguée* de f, définie sur $\mathcal{Y}$ par

$$f^*(y) = \sup \{\langle x, y \rangle - f(x) | x \in \mathcal{X}\}. \tag{1.6}$$

Ainsi f^* est l'enveloppe supérieure des fonctions affines sur $\mathcal{Y}$, $y \longrightarrow \langle x, y \rangle - (x)$ sur $\mathcal{Y}$.

Propriété 1.1.1
On a les propriétés suivantes :

- f^* *est toujours convexe.*
- *Si* f *prend la valeur* $-\infty$ *alors* f^* *est identiquement égale à* $+\infty$.
- *On a* $(f^*)^* \leq f$

On a également la proposition suivante :

Proposition 1.1.1
Soit $f : \mathbb{C} \longrightarrow \overline{\mathbb{R}}$ *alors :*

- $\partial f(x)$ *est une partie convexe fermée de* $\mathcal{Y}$,
- $f \in \Gamma_0(\mathcal{X}) \Longleftrightarrow f^* \in \Gamma_0(\mathcal{Y})$. *Dans ce cas* $f = f^{**}$,
- $y \in \partial f(x) \Longleftrightarrow f(x) + f^*(y) = \langle x, y \rangle$,
- *Si* $f \in \Gamma_0(\mathcal{X})$ *alors* $y \in \partial f(x) \Longleftrightarrow x \in \partial f^*(y)$,
- *Si* $f \in \Gamma_0(\mathcal{X})$ *et si* $\partial f(x)$*est réduit à un singleton* $\{y\}$, *alors* f *est différentiable en* x *et* $\nabla f(x) = y$ *et réciproquement*,
- $f(x^o) = \min \{f(x) | x \in \mathcal{X}\} \Longleftrightarrow 0 \in \partial f(x^0)$.

1.1.3 Fonctions convexes polyédrales

Une partie convexe C est dite convexe polyédrale si

$$C = \bigcap_{i=1}^{m} \{x | \langle a_i, x\rangle - b_i \leq 0, a_i \in \mathcal{Y}, b_i \in \mathbb{R}\}.$$

Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite polyédrale si son épigraphe est un ensemble polyédral de $\mathbb{R}^{n+1}$. Notons que toute fonction polyédrale est propre, convexe et s.c.i.

Proposition 1.1.2
Soit $f : \mathbb{R} \to \bar{\mathbb{R}}$ une fonction convexe. Alors f est polyédrale si et seulement si $dom(f)$ est un ensemble convexe polyédral, et

$$f(x) = \sup\{\langle a_i, x\rangle - b_i | i = 1, 2, \cdots, l\}\, \forall x \in dom(f). \tag{1.7}$$

Proposition 1.1.3

- *Si f est convexe polyédrale alors f^* l'est aussi. De plus si f est partout finie alors*

$$dom(f^*) = co\{a_j | j = 1, 2, \cdots, l\},$$
$$f^*(y) = \min\left\{\sum_{i=1}^{l} \lambda_i a_i | y = \sum_{i=1}^{l} \lambda_i a_i, \lambda_i \geq 0 et \sum_{i=1}^{l} \lambda_i = 1\right\}.$$

- *Si f est polyédrale alors $\partial f(x)$ est une partie convexe polyédrale non vide en tout point de $dom(f)$.*
- *Si $f_1, \cdots, f_s$ sont des fonctions convexes polyédrales sur $\mathcal{X}$ telles que les ensembles convexes $dom(f_i), i = 1, 2, \cdots, s$ ont un point commun alors $\partial(f_1 + \cdots + f_s)(x) = \partial f_1(x) + \cdots + \partial f_s(x)$, $\forall x \in \mathcal{X}$.*

1.1.4 Conditions d'optimalité

Définition 1.1.1
Soient $\mathcal{C} \subset \mathbb{R}^n$ et $x^0 \in \mathbb{R}^n$. On dira que $d \in \mathbb{R}^n$ est tangent à $\mathcal{C}$ en x^0 s'il existe deux suites $\{d_k\}_k \subset \mathbb{R}^n$ et $\{t_k\}_k \subset {\mathbb{R}_+}^n$ telles que

$$d_k \to d, t_k \to 0^+, x^0 + t_k d_k \in \mathcal{C}. \tag{1.8}$$

Proposition 1.1.4
Le cône tangent $\mathcal{T}_{x^0}\mathcal{C}$ est un cône fermé ; il est convexe si $\mathcal{C}$ est convexe.

Considérons le problème d'optimisation

$$(P_{\mathcal{C}}) \begin{cases} \min f(x), \\ x \in \mathcal{C}. \end{cases} \tag{1.9}$$

On a la condition nécessaire d'optimalité suivante

Théorème 1.1.1
Si x^* est un minimum local de f sur $\mathcal{C}$ et si f est dérivable en x^* on a

$$\nabla f(x^*) \in (\mathcal{T}_{x^*}\mathcal{C})^*, \tag{1.10}$$

où $(\mathcal{T}_{x^*}\mathcal{C})^*$ désigne le dual de $\mathcal{T}_{x^*}\mathcal{C}$.
Ce qui se traduit par

$$\nabla f(x^*)d \geq 0, \forall d \in \mathcal{T}_{x^*}\mathcal{C}. \tag{1.11}$$

Considérons à présent le problème d'optimisation suivant

$$(P_{\mathcal{EI}}) \begin{cases} \min f(x), & \\ c_i(x) = 0 & i \in \mathcal{E}, \\ c_i(x) \geq 0 & i \in \mathcal{I}. \end{cases} \tag{1.12}$$

avec $\mathcal{E} = \{1, 2, \cdots m_E\}$ et $\mathcal{I} = \{1, 2, \cdots m_{\mathcal{I}}\}$ définissant une partition de $\{1, 2 \cdots m\}$, $f : \mathbb{R}^n \to \mathbb{R}$, et $c : \mathbb{R}^n \to \mathbb{R}^m$.
On note

$$\mathcal{C} = \{x \in \mathbb{R}^n : c_{\mathcal{E}}(x) = 0, c_{\mathcal{I}}(x) \geq 0\},$$

l'ensemble admissible de $(P_{\mathcal{EI}})$ avec $c_{\mathcal{E}}$ (resp. $c_{\mathcal{I}}$) le vecteur de $\mathbb{R}^{m_E}$ (resp. $\mathbb{R}^{m_{\mathcal{I}}}$) formé des c_i tels que $i \in \mathcal{E}(resp. i \in \mathcal{I})$ et

$$\mathcal{I}^0(x) = i \in \mathcal{I} : c_i(x) = 0,$$

l'ensemble des indices des contraintes d'inégalité actives en x.

Définition 1.1.2
On appelle cône linéarisant de $\mathcal{C}$ en $\overline{x} \in \mathcal{C}$ l'ensemble

$$\mathcal{T}_{\overline{x}}'\mathcal{C} = \left\{d \in \mathbb{R}^m : \nabla_{C\mathcal{E}}(\overline{x})\, d = 0, \nabla_{C_{I^0\overline{x}}} d \geq 0\right\}.$$

On a la proposition suivante

Propriété 1.1.2
si $\overline{x} \in \mathcal{C}$ et si $c_{\mathcal{E}\cup\mathcal{I}^0(\overline{x})}$ est dérivable en $\overline{x}$, alors on a l'inclusion

$$\mathcal{T}_{\overline{x}} \subset \mathcal{T}_{\overline{x}}'. \tag{1.13}$$

Définition 1.1.3
On parle de qualification des contraintes de $(P_{\mathcal{EI}})$ en $\overline{x}$ si on a l'égalité

$$\mathcal{T}_{\overline{x}} = \mathcal{T}_{\overline{x}}'. \tag{1.14}$$

On a le théorème suivant

Théorème 1.1.2
(Condition nécessaire du premier ordre de Karush-Kuhn-Tucker) Soit x^* un minimum local de $(P_{\mathcal{EI}})$. On suppose f et $C_{\mathcal{E}\cup\mathcal{I}^0 x^*}$ dérivables en x^*, on suppose également que les contraintes soient qualifiées en x^*. Alors il existe $\lambda^* \in \mathbb{R}^m$ tel que

$$\begin{cases} \nabla f(x^*) - A(x^*)^T \lambda = & 0, \\ c_{\mathcal{E}}(x^*) = & 0, \\ c_{\mathcal{I}}(x^*) \leq & 0, \\ {\lambda^*}_{\mathcal{I}} \geq & 0, \\ {\lambda^*}_{\mathcal{I}}{}^T c_{\mathcal{I}}(x^*) = & 0, \end{cases}$$

où $A(x^*)$ désigne la jacobienne de c en x^*.

Remarque 1.1.3
Dans le cas convexe, c'est à dire le cas où $c_{\mathcal{E}}$ est affine $c_{\mathcal{I}}$ et f convexes, la condition nécessaire ci-dessus devient une condition suffisante pour x^* d'être minimum(global) de $(P_{\mathcal{EI}})$.

On fait l'hypothèse de la dérivabilité de $c_{\mathcal{E}\cup\mathcal{I}^0}$ en $\overline{x}$ et de la continuité de $c_{\mathcal{I}\,\mathcal{I}^0\overline{x}}$ en $\overline{x}$. La condition de qualification (1.14) étant difficile à vérifier, en pratique on utilise l'une des conditions suivantes :

A. $\mathcal{C}_{\mathcal{E}\cup\mathcal{I}^0\overline{x}}$ est affine dans un voisinage de $\overline{x}$.

B. **Qualification de Slater :**

- $c_{\mathcal{E}}$ est affine avec $\nabla_{c_{\mathcal{E}}}$ surjective,
- $c_{\mathcal{I}^0(\overline{x})}$ est convexe,
- On peut trouver un point $\widehat{x} \in \mathcal{C}$ tel que $c_{\mathcal{I}^0(\widehat{x})\leq 0}$.

C. **Qualification de Mangazarian-Fromowitz :**

si $\sum_{i\in\mathcal{E}\cup\mathcal{I}^0(\overline{x})} v_i \nabla c(\overline{x}) = 0$ avec $v \geq 0$ pour $i \in \mathcal{I}^0(\overline{x})$ alors $v = 0$ pour tout $i \in \mathcal{E} \cup \mathcal{I}^0(\overline{x})$.

1.1.5 Les fonctions DC

Afin d'étendre la programmation convexe à la résolution de la plupart des problèmes d'optimisation non convexes de la vie courante, tout en continuant à utiliser son arsenal théorique et numérique, une nouvelle classe de fonctions est introduite : la classe des fonctions DC. Soit C un ensemble convexe de $\mathcal{X}$. On note $Conv(C)$ l'ensemble des fonctions convexes

sur C à valeur dans $\mathbb{R}$.
Une fonction $f : C \cup \{+\infty\}$ définie sur C est dite DC sur C si elle s'écrit

$$f(x) = g(x) - h(x), \text{ pour tout } x \in C. \tag{1.15}$$

où g et h sont deux fonctions convexes sur C. On note $DC(C)$ l'ensemble des fonctions DC sur C. C'est également l'espace vectoriel engendré par $Conv(C)$. Mise à part sa structure d'espace vectoriel, $DC(C)$ est également stable par rapport aux opérations usuelles utilisées en optimisation.

Proposition 1.1.5 ([3],[4])

- *Une combinaison linéaire de fonctions DC sur C est une fonction DC sur C.*
- *L'enveloppe supérieure(resp. inférieure) d'un nombre fini de fonctions DC à valeurs finies sur C est une fonction DC sur C.*
- *Si f est une fonction DC à valeurs finies sur C alors les fonctions $|f|$, $f^+ = \max\{f, 0\}$ et $f^- = \min\{f, 0\}$ sont des fonctions DC sur C.*

Remarque 1.1.4
Soit $f \in DC(C)$ et soit $f = g - h$ sa représentation DC. Pour toute fonction convexe finie ξ sur C, $f = (g + \xi) - (h + \xi)$ définit une autre décomposition DC de f. Ainsi une fonction DC possède une infinité de décompositions DC.

Une sous-classe importante de fonctions de $Conv(C)$ est la classe de fonctions C^2 sur $\mathbb{R}^n$. Cette classe est particulièrement importante car les fonctions objectif de la plupart des problèmes d'optimisation de la vie courante appartiennent à cette sous-classe.

On note $C^{1,1}$ le sous-espace vectoriel de $C^1(C)$ des fonction dont le gradient est localement Lipschitzien sur C. $C^1(C)$ est contenu dans le cône convexe $\mathcal{LC}^2(C)$ des fonctions localement enveloppe supérieure d'une famille de fonctions de $C^2(C)$. Ces fonction sont dites sous $-C^2$.

Définition 1.1.4
Soit C un ouvert convexe, f est localement DC sur C si tout point x^0 de C admet un voisinage $\mathcal{V}(x^0)$ noté $\mathcal{V}$, tel que pour tout x dans $\mathcal{V}$

$$f(x) = g_{\mathcal{V}}(x) - h_{\mathcal{V}}(x) \text{ pour tout } g_{\mathcal{V}}, h_{\mathcal{V}} \in Conv(\mathcal{V}).$$

Proposition 1.1.6 ([3], [17])
Les deux propositions suivantes sont équivalentes

i *f est dans $\mathcal{LC}^2(C)$*

ii *f est localement DC sur C avec $h_{\mathcal{V}}$ est une forme quadratique convexe.*

On a le théorème suivant,

Théorème 1.1.3 ([3], [17])
Soit $\mathcal{C}$ un ouvert convexe de $\mathcal{X}$.

- *Toute fonction localement DC sur C est DC sur C. En particulier toute fonction de classe C^2 sur C est DC sur C.*
- *On a la chaîne d'inclusions*
$$Conv\,(C) \subset \mathcal{LC}^2(C) \subset DC\,(C)\,.$$
- *Si C est en plus compact, alors $DC\,(C)$ est un sous-espace vectoriel dense dans l'ensemble des fonctions continues sur C muni de la norme de la convergence uniforme sur C.*

1.1.6 Dualité en optimisation DC

Soient g et h deux fonctions convexes propres et sci sur $\mathcal{X}$. Considérons le problème d'optimisation

$$(P_{dc})\ \alpha = \inf\{g(x) - h(x) | x \in \mathcal{X}\}.$$

Puisque h est dans $\Gamma_0(X)$ on a $h^{**} = h$, on peut donc écrire

$$h(x) = \sup\{\langle x, y\rangle - h^*(y) | y \in \mathcal{Y}\}\,,$$

et on a

$$\begin{aligned} \alpha &= \inf\{g(x) - \sup\langle x, y\rangle - h^*(y) | y \in \mathcal{Y} | x \in \mathcal{X}\} \\ &= \inf\{\beta(y) | y \in \mathcal{Y}\} \end{aligned}$$

avec

$$\beta(y) = \inf\{g(x) - [\langle x, y\rangle - h^*(y)]\,| x \in \mathcal{X}\}\,.$$

Comme (P_y) est un problème convexe et

$$\beta(y) \begin{cases} h^*(y) - g^*(y) si y \in dom(h^*), \\ -\infty\ sinon. \end{cases}$$

On obtient finalement le problème dual de (P_{dc})

$$\alpha = \inf\{h^*(y) - g^*(y) | y \in dom(h^*)\}\,,$$

et grâce à la convention (1.1) on peut écrire

$$(D_{dc})\ \alpha = \inf\{h^*(y) - g^*(y) | y \in \mathcal{Y}\}\,.$$

On observe la parfaite symétrie entre le problème primal (P_{dc}) et le problème dual (D_{dc}) ; il va de soi que les résultats établis pour l'un se transposent à l'autre.

1.1.7 Conditions d'optimalité en optimisation DC

Soient $\mathcal{S}_P$ et $\mathcal{S}_D$ les ensembles de solutions des prolèmes (P_{dc})et(D_{dc}) respectivement, et posons

$$\mathcal{P}_l = \{x^* \in \mathcal{X} | \partial h(x^*) \subset \partial g(x^*)\} \, et \mathcal{D}_l = \{y^* \in \mathcal{Y} | \partial g^*(x^*) \subset \partial h^*(x^*)\}.$$

Le théorème suivant est celui à partir duquel DCA est bâti.

Théorème 1.1.4 ([8], [4])

(i) Transport de minima globaux :

$$\cup \{\partial h(x) | x \in \mathcal{S}_P\} \subset \mathcal{S}_D \subset dom(h^*).$$

La première inclusion se transforme en égalité si g^* est sous-différentiable sur S_D(en particulier si $S_D \subset ir(dom(g^*))$ où si g^* est sous-différentiable sur $dom(h^*)$) et dans ce dernier cas $\mathcal{S}_D \subset (dom(\partial g^*) \cap dom(\partial h^*))$.

(ii) Si x^* est un minimum local de $g - h$, alors $x^* \in \mathcal{P}_l$. L'implication inverse est vraie si h est une fonction convexe polyédrale.

(iii) Soit x^* un point critique[2] de $g - h$ et $y^* \in \partial g(x^*) \bigcap \partial h(x^*)$. Soit $\mathcal{U}$ un voisinage de x^*tel que $\mathcal{U} \bigcap dom(g) \subset dom(\partial h)$. Si pour tout $x \in \mathcal{U} \bigcap dom(g)$ il existe un $y \in \partial h(x)$ tel que $h^*(y) - g^*(y) \geq h^*(y^*) - g^*(y^*)$, alors x^* est un minimum local de $g - h$. Plus précisément,

$$g(x) - h(x) \geq g(x^*) - h(x^*), \text{ pour tout } x \in \mathcal{U} \bigcap dom(g). \tag{1.16}$$

(iv) Transport de minima locaux : soit $x^* \in dom(\partial h)$ un minimum local de $g - h$ et soit $y^* \in \partial h(x^*)$. Sous l'hypothèse

$$y^* \in int(dom(g^*)) et \partial g^*(y^*) \subset \mathcal{U},$$

par exemple si g^* est différentiable en y^*, alors y^* est un minimum local de $h^* - g^*$.

Notons que ce théorème admet sa forme duale grâce à la symétrie observée à la section précédente entre le problème (P_{dc}) et le problème (D_{dc}). Notons également que tous ces résultats concernent les composantes DC g et h et non la fonction f elle-même.

Propriété 1.1.3 ([8], [4])

Soit $f = g - h$ avec $g, h \in \Gamma_0(\mathcal{X})$ vérifiant $dom(g) \subset dom(h)$ et $ir(dom(g)) \cap ir(dom(h)) \neq \oslash$. Soit $x_o \in dom(g)(resp.dom(h))$ un point où g $(resp.h)$ est continue. Si f est convexe, alors

(i) h est continue en x^o,

2. *x^* est un point critique de $g - h$ si $\partial h(x^*) \cap \partial g(x^*)$* est non vide.

(ii) $\partial h(x^{\circ}) = \partial g(x^o) * \partial h(x^o)$[3], [3]

(iii) $0 \in \partial f(x^o) \Longleftrightarrow \partial h(x^o) \subset \partial g(x^o)$.

Preuve : On a dom(f) = dom(g)et $g = f + h$. Donc $\partial g(x^o) = \partial f(x^o) + \partial h(x^o)$ car $ir(dom(f)) \cap ir(dom(h)) \neq \oslash$. Si g est continue en $x^o \in dom(g)$, on a

$$x^o \in ir(dom(g)) = int(dom(g)) \subset int(dom(h)). \tag{1.17}$$

Ainsi, l'ensemble convexe fermé $\partial h(x^o)$ est borné et de [5] on a

$$\partial f(x^o) = \partial g(x^o) * \partial h(x^o) = \bigcap \{\partial g(x^o) - v | v \in \partial h(x^o)\} \tag{1.18}$$

La troisième équivalence est immédiate. On observe donc que les problèmes d'optimisation convexes peuvent s'écrire comme problème d'optimisation DC et résolus en utilisant les techniques d'optimisation DC.
Soient $f, \theta \in \Gamma_0$ telles que $dom(f) \subset dom(\theta)$ *et* $ir(dom(f)) \cap ir(dom(\theta)) \neq \oslash$ alors le problème d'optimisation

$$\inf \{f(x) | x \in \mathcal{X}\},$$

est équivalent au faux problème d'optimisation DC

$$\inf \{f(x + \theta) - \theta(x) | x \in \mathcal{X}\},$$

dans le sens où ils ont la même valeur optimale et le même ensemble de solutions. On a

$$0 \in \partial f(x^*) \Longrightarrow \partial\theta(x^*) \subset \partial(f + \theta)(x^*) = \partial f(x^*) + \partial\theta(x^*),$$

et l'autre implication est vraie si, en plus, θ est continue en x^*.

Propriété 1.1.4 ([8], [4])
Soit $f = g - h$ avec $g, h \in \Gamma_0(\mathcal{X})$ vérifiant $dom(g) \subset dom(h)$. S'il existe un voisinage convexe $\mathcal{U}$ de x^* tel que f est finie et convexe sur $\mathcal{U}$, alors les propositions suivantes sont équivalentes :

(i) $0 \in \partial(f + \mathcal{X}_{\mathcal{U}})(x^*)$,

(ii) $\partial h(x^*) \subset \partial g(x^*)$.

Sous les hypothèses ci-dessus, la condition nécessaire d'optimalité locale du théorème (1.1.4) ;

$$\partial h(x^*) \subset \partial g(x^*),$$

est également suffisante : $f(x^*) \leq f(x)$ pour tout $x \in \mathcal{U}$.

Théorème 1.1.5 ([5], [4], [18])
Soit $f = g - h$ où $g, h \in \Gamma_0(\mathcal{X})$ alors x^* est minimum global du problème (P_{dc}) si et seulement si

$$\partial_\epsilon h(x^*) \subset \partial_\epsilon g(x^*) \textit{pour tout } \epsilon > 0. \tag{1.19}$$

3. $A * B = \{x \in \mathcal{X} | x + B \subset A\}$ [5].

1.1.8 Algorithme d'optimisation DC : DCA

La construction des DCA repose sur la génération de deux suites $\left\{x^k\right\}_k$ et $\left\{y^k\right\}_k$ qui sont améliorées à chaque itération de sorte que leur limite respective x^* et y^* soient candidates pour être les optima locaux du problème primal et du problème dual respectivement. Ces deux suites sont liées par la dualité et vérifient les propriétés suivantes :

i. Les suites $g(x^k) - h(x^k)$ et $g^*(y^k) - h^*(y^k)$ décroissent à chaque itération,

ii. Si $(g-h)(x^{k+1}) = (g-h)(x^k)$, l'algorithme s'arrête à l'itération $k+1$, et le point $x^k(resp.y^k)$ est un point critique de $g-h$ (resp. h^*-g^*),

iii. sinon toute valeur d'adhérence x^* de la suite $\left\{x^k\right\}_k (resp.y^*$ de la suite $\left\{y^k\right\})$ est un point critique de $g-h(resp.h^*-g^*)$.

Ces suites sont générées de la manière suivante : x^{k+1} $(resp.y^k)$ est solution du problème convexe (P_k) $(resp.(D_k))$ défini par

$$(P_k)\alpha_k = \inf\left\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] | x \in \mathcal{X}\right\}. \tag{1.20}$$

$$(D_k)\inf\left\{h^*(y) - [g^*(y^{k-1}) + \langle x^k, y - y^{k-1}\rangle] | y \in \mathcal{Y}\right\}. \tag{1.21}$$

Comme on peut le voir (P_k) $(resp.(D_k))$ est obtenu de $(P_{dc})(resp.(D_{dc}))$ en remplaçant $h(resp.g^*)$ par sa minorante affine définie par $y^k \in \partial h(x^k)(resp.x^k \in \partial g^*(y^{k-1}))$, DCA conduit au schéma

$$\begin{array}{ccc} x^k & \longrightarrow & y^k \in \partial h(x^k) \\ & \swarrow & \\ x^{k+1} \in \partial g^*(y^k) & \longrightarrow & y^{k+1} \in \partial h(x^{k+1}) \end{array}$$

Ce schéma correspond à la forme simplifiée de DCA, et se traduit par l'algorithme ci-dessous.

Algorithme 1.1.1
DCA forme simplifiée

1. x^0 *donné,*
2. *Pour chaque* k, x^k *étant connu, déterminer* $y^k \in \partial h(x^k)$,
3. *Trouver* $x^{k+1} \in \partial g^*(y^k)$,
4. *Si test d'arrêt vérifié Stop ; sinon* $k \longleftarrow k+1$ *et aller en 1.*

Dans la forme complète de DCA on impose le choix suivant :

$$x^{k+1} \in arg\ min\left\{g(x) - h(x) : x \in \partial g^*(y^k)\right\}, \tag{1.22}$$

et

$$y^k \in arg\ min\left\{h^*(y) - g^*(y) : y \in \partial h(x^k)\right\}. \tag{1.23}$$

Les problèmes (1.22) et (1.23) sont équivalents aux problèmes respectifs

$$x^{k+1} \in arg\ min\left\{\langle x, y^k\rangle - h(x) : x \in \partial g^*(y^k)\right\}, \tag{1.24}$$

et

$$y^k \in arg\ min\left\{\langle x^k, y\rangle - g^*(y) : y \in \partial h(x^k)\right\}. \tag{1.25}$$

Les problèmes (1.24) et (1.25) sont des problèmes de minimisation concave. Ainsi DCA assure l'appartenance $(x^\infty, y^\infty) \in \mathcal{P}_l \times \mathcal{D}_l$

Malgré leur apparente simplicité par rapport aux problèmes d'origine (P_{dc})et(D_{dc}), ces problèmes restent difficiles à résoudre. En effet il s'agit de problèmes de minimisation concave comme nous l'avons déjà signalé plus haut, de plus le travail se fait sur $\partial g^*(y^k)$ (resp. $\partial h(x^k)$). Ces difficultés rendent en pratique l'utilisation de ce schéma presque impossible, excepté pour des cas bien particuliers pour lesquels la résolution de (1.24)-(1.25) est une tâche facile. Dans la suite DCA sera utilisé pour désigner DCA simplifié, sauf indication contraire.

1.1.9 Existence et bornitude des suites générées par DCA

L'algorithme DCA est bien défini si on peut effectivement construire les deux suites $\left\{x^k\right\}_k$ et $\left\{y^k\right\}_k$ comme ci dessus à partir d'un point initial arbitraire x^0.

Lemme 1.1.1 ([4], [8])
Les suites $\left\{x^k\right\}_k$, $\left\{y^k\right\}_k$ dans DCA sont bien définies si et seulement si

$$dom(\partial g) \subset dom(\partial h), et dom(\partial h^*) \subset dom(\partial g^*).$$

La proposition suivante établit les conditions pour lesquelles les suites générées par DCA sont bornées.

Lemme 1.1.2 ([4], [8])
Si $g - h$ est coercive[4] *alors on a*

(i) la suite $\left\{x^k\right\}_k$ est bornée,

(ii) si $\left\{x^k\right\}_k \subset int\,(dom\,(h))$ alors la suite $\left\{y^k\right\}_k$ est aussi bornée.

Par dualité, si $(h^* - g^*)$ est coercive alors on a

(i) La suite $\left\{y^k\right\}_k$ est bornée,

(ii) Si $\left\{y^k\right\}_k \subset int\,(dom\,(g^*))$ alors $\left\{x^k\right\}_k$ est aussi bornée.

4. une fonction Ψ est coercive si $\lim_{\|x\|\to+\infty} \Psi(x) = +\infty$

Soient ρ_i et ${\rho^*}_i$, (i=1, 2) des nombres réels positifs tels que $0 \leq \rho_i < \rho(f_i)$
(resp. $0 \leq {\rho^*}_i < {\rho^*}_i({f^*}_i)$) où $\rho_i = 0(resp\ {\rho^*}_i = 0)$ si $\rho(f_i) = 0$ (resp $\rho({f^*}_i = 0)$) et ρ_i (resp ${\rho^*}_i$) peut prendre la valeur $\rho(f_i)$ (resp $\rho({f^*}_i)$) si cette borne supérieure est atteinte. Nous poserons pour la suite $f_1 = g$, $f_2 = h$.
Le théorème suivant établit la convergence de DCA simplifié.

Théorème 1.1.6 ([4], [8])
Si les suites $\left\{x^k\right\}_k$ et $\left\{y^k\right\}_k$ sont bien définies. Alors on a :

i $(g-h)(x^{k+1}) \leq (h^*-g^*)(y^k) - \frac{\rho_2}{2}\left\|dx^k\right\|^2 \leq (g-h)(x^k) - \frac{\rho 1+\rho 2}{2}\left\|dx^k\right\|^2$,

ii $(h^*-g^*)(y^{k+1}) \leq (g-h)(x^{k+1}) - \frac{{\rho_1}^*}{2}\left\|dy^k\right\|^2 \leq (h^*-g^*)(y^k) - \frac{\rho 1^*+\rho 2^*}{2}\left\|dy^k\right\|^2$,
où $dx^k = x^{k+1} - x^k$.

Corollaire 1.1.1

1.

$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^*-g^*)(y^k) - \frac{\rho_2}{2}\left\|dx^k\right\|^2 \\ &\leq (g-h)(x^k) - \left[\frac{\rho_2}{2}\left\|dx^{k-1}\right\|^2 + \frac{\rho_1^*}{2}\left\|dy^k\right\|^2\right].\end{aligned}$$

2.

$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^*-g^*)(y^k) - \frac{\rho_2^*}{2}\left\|dx^k\right\|^2 \\ &\leq (g-h)(x^k) - \left[\frac{\rho_2^*}{2}\left\|dx^{k-1}\right\|^2 + \frac{{\rho_1}^*}{2}\left\|dy^k\right\|^2\right].\end{aligned}$$

3.

$$(h^*-g^*)(y^{k+1}) \leq (g-h)(x^{k+1}) - \frac{{\rho_1}^*}{2}\left\|dy^k\right\|^2 \ \leq (h^*-g^*)(y^k - \left[\frac{{\rho_1}^*}{2}\left\|dy^k\right\|^2 + frac{\rho_2}^*2\left\|dx^k\right\|^2\right].$$

4.

$$(h^*-g^*)(y^{k+1}) \leq (g-h)(x^{k+1}) - \frac{\rho_1}{2}\left\|dy^{k+1}\right\|^2 \ \leq (h^*-g^*)(y^k) - \left[\frac{\rho_1}{2}\left\|dx^{k+1}\right\|^2 + \frac{\rho_2}{2}\left\|dy^{k+1}\right\|^2\right].$$

Corollaire 1.1.2
Si les égalités ont lieu, alors :

1. $(g-h)(x^{k+1}) = (h^*-g^*)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1})$.
2. $(g-h)(x^{k+1}) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^*(y^k), y^k \in \partial h(x^{k+1}$.
3. $(h^*-g^*)(y^k) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^*$.

4. $(h^* - g^*)(y^{k+1}) = (h^* - g^*)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1}), x^{k+1} \in \partial g^*(y^{k+1})$.

En général, les qualités (robustesse, stabilité, vitesse de convergence, bonnes solutions locales) de DCA dépendent des décompositions DC de la fonction objectif $f = g - h$. Le théorème1.1.6 montre que la forte convexité des composantes convexes dans le problème primal et dual peut influencer sur DCA. Pour rendre les composantes convexes g et h fortement convexes, on peut usuellement appliquer l'opération suivante

$$f = g - h = \left(g + \frac{\lambda}{2}\|.\|^2\right) - \left(h + \frac{\lambda}{2}\|.\|^2\right).$$

Dans ce cas, les composantes convexes dans le problème dual seront continûment différentiables

1.1.10 Optimisation DC polyédrale : étude approfondie de DCA

On parle d'optimisation DC polyédrale lorsque l'une des composantes convexes de la décomposition DC $f = g - h$ est convexe polyédrale. Cette classe de problèmes DC se rencontre fréquemment dans la pratique et possède des propriétés intéressantes tant sur le plan théorique que numérique. En particulier DCA a une convergence finie [[4], [8]]. Elle nous permet également de donner une interprétation profonde de DCA. Notons $h_k(resp.h^k)$ la minorante affine(resp. convexe polyédrale) de la fonction convexe h.

$$\begin{aligned} h_k(x) &= h(x^k) + \langle x - x^k, y^k\rangle. \\ &= \langle x, y^k\rangle - h^*(y^k), \forall x \in \mathcal{X} \end{aligned}$$

$$\begin{aligned} h^k(x) &= sup\left\{h_i(x) | i = 0, 1, \cdots, k\right\} \\ &= sup\left\{\langle x, y^i\rangle - h^*(y^i) | i = 0, 1, \cdots, k\right\}, \forall x \in \mathcal{X} \end{aligned}$$

où les suites $\left\{x^k\right\}_k$ et $\left\{y^k\right\}$ sont obtenues en suivant la procédure décrite à la Section 1.1.8. On définit de manière duale $(g^*)_k$(resp. $(g^*)^k$) la minorante affine(resp. convexe polyédrale) de la fonction convexe g^* :

$$\begin{aligned} {g^*}_k(x) &= g^*(y^{k-1}) + \langle y - y^{k-1}, x^k\rangle. \\ &= \langle y, x^k\rangle - g(x^k), \forall y \in \mathcal{Y} \end{aligned}$$

$$\begin{aligned} (g^*)^k(x) &= sup\left\{(g^*)_i(y) | i = 0, 1, \cdots, k\right\} \\ &= sup\left\{\langle y, x^i\rangle - g\left(x^i\right) | i = 0, 1, \cdots, k\right\}, \forall x \in \mathcal{X} \end{aligned}$$

Etant donné qu'une fonction convexe propre s.c.i se caractérise comme étant le suprémum de ses minorantes affines, il s'avère donc plus judicieux d'utiliser h_k(resp. $(g^*)_k$) comme sous estimé la fonction convexe h(resp g^*), au lieu de la minorante affine h_k(reps. $(g^*)_k$). On obtient alors les programmes

$$\inf\left\{g(x) - h^k(x) | x \in \mathcal{X}\right\}, \tag{1.26}$$

$$\inf\left\{h^*(y) - (g^*)^k(y) | y \in \mathcal{Y}\right\}. \tag{1.27}$$

Ces programmes sont non convexes en opposition aux sous-problèmes (P_k) et (D_k), qui eux, l'étaient.

Lemme 1.1.3
x^{k+1} est une solution optimale du problème (1.26)

Preuve
on commence tout d'abord par réécrire (1.26) sous la forme,

$$\inf\left\{g(x) - \sup h_i(x) | i = 0, 1, \cdots, k | x \in \mathcal{X}\right\}, \tag{1.28}$$

qui peut également s'écrire

$$\inf_{x\in\mathcal{X}} \inf_{i=1\cdots k} \left\{g(x) - h_i(x)\right\} = \inf_{i=1\cdots k} \inf_{x\in\mathcal{X}} \left\{g(x) - h_i(x)\right\}. \tag{1.29}$$

Puisque x^{i+1} est la solution optimale du problème (P_i), on déduit que x^i avec

$$i \in arg\ min\left\{g(x^{i+1}) - h_i(x^{i+1}) | i = 0, 1, \cdots, k\right\},$$

est une solution optimale du problème (1.26).
D'autre part on a

$$g(x^{i+1}) - h(x^{i+1}) \leq\ g(x^{i+1}) - h_i(x^{i+1}),\ pour\ i\ = 0, 1, \cdots, k. \tag{1.30}$$

Puisque la suite $\left\{g(x^i) - h(x^i)\right\}_i$ est décroissante on a

$$g(x^{k+1}) - h(x^{k+1}) \leq inf g(x^{i+1}) - h_i(x^{i+1}),\ pour\ i\ = 0, 1, \cdots, k.$$

Et si

$$h_k(x^{k+1}) = h(x^{k+1}), \tag{1.31}$$

alors $h^k(x^{k+1}) = h(x^{k+1})$ et on obtient l'égalité

$$g(x^{k+1}) - h(x^{k+1}) = g(x^{k+1}) - h_k(x^{k+1}) = inf g(x^{i+1}) - h_i(x^{i+1}),\ pour\ i = 0, 1, \cdots, k.$$

Donc x^{k+1} est une solution optimale du problème DC polyédrale(1.26). Le lemme suivant apporte les mêmes précisions pour le problème (1.27).

Lemme 1.1.4 ([4], [8])
y^k est une solution optimale du problème (1.27).

Preuve
Le schéma de la démonstration est similaire à celui du Lemme (1.1.3) en utilisant cette fois les expressions duales.
Dans la démonstration du Lemme 1.1.3 nous avons fait l'hypothèse que l'égalité $h_k(x^{k+1}) = h(x^{k+1})$ est vraie. Qu'en est-il si se n'est pas le cas ? Plus précisément que se passe-t-il si

$$h_k(x^{k+1}) < h(x^{k+1}) \text{ pour tout } k \text{ ?} \tag{1.32}$$

Ce cas correspond à celui où k tend vers l'infini dans (1.31). Nous allons décrire dans les lignes qui suivent le comportement de DCA dans ce cas. Supposons que la valeur optimale du problème (P_{dc}) soit finie et les suites $\left\{x^k\right\}_k$ et $\left\{y^k\right\}_k$ générées par DCA. Alors

$$g(x^\infty) - h_\infty(x^\infty) = inf\left\{g(x^{i+1}) - h_i(x^{i+1}) | i = 0, 1, \cdots, \infty\right\} \tag{1.33}$$

pour toute valeur d'adhérence x^∞ de la suite $\left\{x^k\right\}_k$, où h_∞ est la minorante affine de h :

$$h_\infty(x) = h(x^\infty) + \langle x - x^\infty, y\rangle = \langle x, y^\infty\rangle - h^*(y^\infty), \forall x \in \mathcal{X}. \tag{1.34}$$

avec $y^\infty \in \partial h(x^\infty)$ une valeur d'adhérence de $\left\{y^k\right\}_k$. Le point x^∞ est par conséquent une solution du programme DC.

$$\inf\left\{g(x) - h^\infty(x) | x \in \mathcal{X}\right\}, \tag{1.35}$$

où la fonction convexe h^∞ est définie par

$$h^\infty(x) = sup\left\{\langle x, y^i\rangle - h^*(y^i) | i = 0, 1, \cdots, 1\right\}, \forall x \in \mathcal{X}.$$

De manière similaire pour le problème dual DC (D_{dc}), une minorante affine de g^* est définie par

$$(g^*)_\infty(y) = (g^*)_\infty(y^\infty) + \langle y - y^\infty, x^\infty\rangle = \langle y, x^\infty\rangle - g(x^\infty), \forall x \in \mathcal{X}$$

et

$$(g^*)_\infty(y) = sup(g^*)_i(y) | i = 1, 2, \cdots, 1, \forall y \in \mathcal{Y}. \tag{1.36}$$

Et comme plus haut on a

$$h^*(y^\infty) - (g^*)^\infty(y^\infty) = inf\left\{h^*(y^i) - (g^*)_i(y^i) | i = 1, 2, \cdots, 1\right\} \tag{1.37}$$

et y^∞ est une solution du problème DC

$$\inf\left\{h^*(y) - (g^*)^\infty(y) | y \in \mathcal{Y}\right\}. \tag{1.38}$$

De plus les valeurs optimales des problèmes (1.35) et (1.38) sont égales

$$g(x^\infty) - h^\infty(x^\infty) = h^*(y^\infty) - (g^*)(y^\infty). \tag{1.39}$$

On a les théorèmes fondamentaux suivants.

Théorème 1.1.7 ([4], [8])

(i) $g(x^{k+1}) - h(x^{k+1}) = h^*(y^k) - g^*y^k)$ *si et seulement si* $h^k(x^{k+1}) = h(x^{k+1})$,

(ii) $h^*(y^k) - g^*(y^k) = g(x^k) - h(x^k)$ *si et seulement si* $g^*(y^k) = (g^*)^k(y^k)$,

(iii) $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *si et seulement si* $h^k(x^{k+1}) = h(x^{k+1}) et g^*(y^k) = (g^*)^k(y^k)$. *Par conséquent, si* $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *(dans ce cas la valeur commune est* $h^*(y^k) - g^*(y^k)$*), alors les propositions suivantes sont vraies,*

(iv) x^{k+1} *(resp.* y^k*) est une solution du problème (1.26) (resp. (1.27).*

(v) *De plus, si h et* h^k *coïncident en une solution de* (P_{dc}) *ou* g^* *et* $(g^*)^k$ *coïncident en une solution de* (D_{dc})*, alors* x^{k+1} *(resp.* y^k*) est également une solution de* (P_{dc}) *(resp.* (D_{dc})*.*

Théorème 1.1.8

Si la valeur optimale du problème DC (P_{dc}) *est finie et les suites* $\{x^k\}_k$ *et* $\{y^k\}_k$ *sont bornées, alors toute valeur d'adhérence* x^∞*(resp.* y^∞*) est une solution du problème DC (1.38)(resp.(1.35)). De plus les valeurs optimales sont égales,*

$$g(x^\infty) - h^\infty(x^\infty) = h^*(y^\infty) - (g^*)^\infty(y^\infty).$$

Si l'une des conditions ci-dessous est vérifiée,

(i) *les fonctions* h^∞ *et h coïncident en une solution optimale de* (P_{dc}),

(ii) *les fonctions* $(g^*)^\infty$ *et* g^* *coïncident en une solution optimale de* (D_{dc})*, alors* x^∞ *et* y^∞ *sont également des solutions optimales respectivement de* (P_{dc}) *et* (D_{dc})*.*

1.2 Un exemple test

Dans cette section pour illustrer le processus de convexification qu'entreprend DCA pendant ses itérations et l'influence qu'a le point initial et le choix de la décomposition DC sur la qualité de la solution que trouve DCA, nous considèrerons le problème d'optimisation quadratique non convexe suivant généré aléatoirement,

$$(\mathcal{T}) = \left\{ \begin{array}{l} \min f(x,y) = \frac{1}{2}(x,y) \begin{pmatrix} -83.75 & 28.34 \\ 28.34 & -48.24 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ t.q \\ 0 \leq x \leq 1, 0 \leq y \leq 1. \end{array} \right\}.$$

Ce problème a quatre solutions locales (0, 0), (1, 0), (0, 1) et (1, 1) comme le montre la figure(1.3) La solution globale est (1, 0) avec comme valeur objectif -24.20.
La plus petite valeur propre du Hessien de f est -99.43. La décomposition DC de la fonction objectif considérée pour cet exemple est la suivante $f(x,y) = g(x,y) - h(x,y)$ avec

$$g(x,y) = \frac{1}{2}(x,y) \begin{pmatrix} 15.68 & 28.34 \\ 28.34 & 51.19 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$h(x,y) = \tfrac{1}{2}(x,y)\begin{pmatrix} 99.43 & 0 \\ 0 & 99.43 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Une décomposition alternative est $f(x,y) = \overline{g}(x,y) - \overline{h}(x,y)$ avec

$$\overline{g}(x,y) = \tfrac{1}{2}(x,y)\begin{pmatrix} 99.43 & 0 \\ 0 & 99.43 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

$$\overline{h}(x,y) = \tfrac{1}{2}(x,y)\begin{pmatrix} 183.18 & -28.34 \\ -28.34 & 147.67 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

1.2.1 Importance du choix du point initial pour DCA

Le tableau (1.1) présente les différentes solutions trouvées par DCA en fonction des différents points initiaux. Ces points sont obtenus à partir d'une perturbation de la solution globale. On voit sur le tableau (1.1) l'intérêt de fournir à DCA un bon point initial. En effet

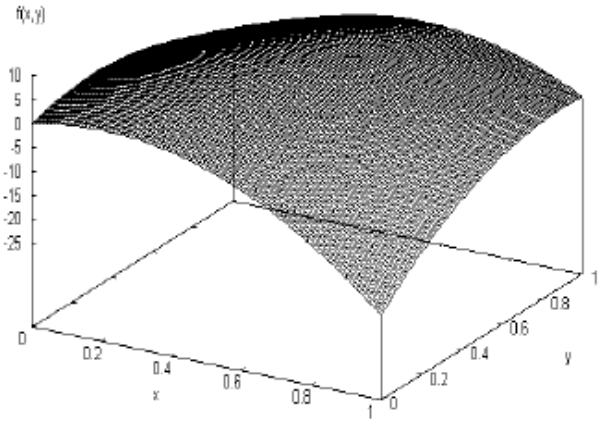


FIGURE 1.3 – La surface de $f\,(x,y)$.

en gras, la composante y du deuxième point ne diffère de la composante y du premier point que de 0.01 la composante x des deux points étant identique, on observe la grande différence des valeurs objectif trouvées par DCA dans les deux cas.

(x^0,y^0)	$iter$	f^*	(x^*,y^*)
(0.75,0.25)	2	-24.20	(1.00,0.00)
(0.32,0.25)	3	-24.20	(1.00,0.00)
(0.24,0.25)	3	0.00	(0.00,0.00)
(0.24,0.27)	4	0.00	(0.00,0.00
(0.24,0.31)	8	**0.00**	**(0.00,0.00)**
(0.24,0.32)	9	**-8.92**	**(0.00,1.00)**
(0.25,0.75)	2	-8.92	(0.00,1.00)

TABLE 1.1 – DCA avec différents points initiaux.

Le choix d'un bon point initial pour DCA est une tâche difficile et cette question reste ouverte. Cependant un début de réponse semble être l'adjonction à DCA d'une procédure pour calculer un bon point initial.
Notons qu'en utilisant la fonction **quadprog** de MATLAB pour résoudre le problème $(\mathcal{T})$ et en prenant comme points initiaux les mêmes que ceux indiqués sur le tableau (1.1), la solution globale (1, 0) est trouvée à chaque fois. Aucune conclusion sur la supériorité de quadprog par rapport à DCA ne peut néanmoins être tirée ; cependant cette remarque ne fait que souligner l'importance particulière que revêt le choix du point initial pour DCA.

1.2.2 Processus de convexification de DCA

La figure (1.4) présente le processus de convexification qu'entreprend DCA. Comme on peut le voir, et comme cela a été présenté plus haut, la convexification de DCA est globale et majorante. Elle est globale car elle se fait sur tout le domaine admissible, et elle est majorante car comme le montre la figure (1.4), la fonction DC $f = g-h$ est remplacée à chaque itération k par la fonction convexe majorante $\overline{f}_k = g - \langle ., y^k\rangle$ avec $y^k \in \partial h(x^k)$.

1.2.3 Importance de la décomposition DC pour DCA

La qualité de la solution retournée par DCA dépend également de la convexification qu'entreprend DCA et celle-ci dépend de la décomposition DC adoptée. Pour l'illustrer considérons une fois de plus le problème $(\mathcal{T})$ et considérons une fois de plus la décomposition DC $f(x,y) = g(x,y) - h(x,y)$ avec

$$g(x,y) = \tfrac{1}{2}(x,y)\begin{pmatrix} 15.68+\varsigma & 28.34 \\ 28.34 & 51.19+\varsigma \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

$$h(x,y) = \tfrac{1}{2}(x,y)\begin{pmatrix} 99.43+\varsigma & 0 \\ 0 & 99.43+\varsigma \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

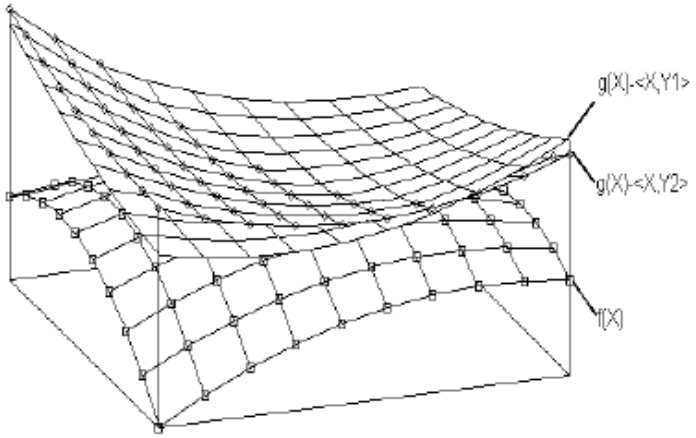


FIGURE 1.4 – Processus de convexification de DCA.

où ς est une perturbation positive que nous faisons varier au Tableau (1.2) pour montrer qu'un choix de ς petit entraîne un meilleur rapport qualité de la solution nombre d'itérations donc temps de calcul pour DCA. Le point initial est (0.320, 0.25) dans cette analyse.

(ς	$iter$	f^*	(x^*, y^*)
0.001	3	-24.20	(1.00,0.00)
0.01	3	-24.20	(1.00,0.00)
0.1	3	-24.20	(0.00,0.00)
1	4	-24.20	(0.00,0.00
10	3	-24.20	(1.00,0.00)
20	4	-24.20	(1.00,0.00)
25	4	0.00	(0.00,0.00)
30	6	0.00	(0.00,0.00)

TABLE 1.2 – DCA avec différentes décompositions DC

Chapitre 2

Les algorithmes génétiques

2.1 Introduction

Les algorithmes génétiques (AG) appartiennent à la famille des algorithmes évolutionnistes (un sous-ensemble des méta heuristiques [19]). Leur but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation, lorsqu'il n'existe pas (ou qu'on ne connaît pas) de méthode exacte pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle développée au XIX^{me} siècle par le scientifique Darwin et l'appliquent à une population de solutions potentielles au problème donné.

Les algorithmes génétiques ont la particularité de s'inspirer de l'évolution des espèces dans leur cadre naturel. Les espèces s'adaptent à leur cadre de vie qui peut évoluer, les individus de chaque espèce se reproduisent, créant ainsi de nouveaux individus, certains subissent des modifications de leur ADN, certains disparaissent....

Un algorithme génétique va reproduire ce modèle d'évolution dans le but de trouver des solutions pour un problème donné. On fera usage, alors, de termes empruntés au monde des biologistes et des généticiens et ceci afin de mieux représenter chacun des concepts abordés :

1- Dans notre cas, une population sera un ensemble d'individus.

2- Un individu sera une réponse à un problème donné, qu'elle soit ou non une solution valide du problème.

3- Un gène sera une partie d'une réponse au problème, donc d'un individu.

4- Une génération est une itération de notre algorithme.

Un algorithme génétique va faire évoluer une population dans le but d'en améliorer les individus. Et c'est donc, à chaque génération, un ensemble d'individus qui sera mis en avant et non un individu particulier. Nous obtiendrons donc un ensemble de solutions pour un problème et pas une solution unique. Les solutions trouvées seront généralement différentes, mais seront d'une qualité équivalente.

Dans ce chapitre, On va commencer par donner le principe des algorithmes génétiques en détaillant les différents paramètres utiles pour l'implémentation de l'approche et on va terminer par citer quelques avantages et inconvénients de ces algorithmes.

2.2 Principes généraux

Le but des algorithmes génétiques est de déterminer les extrêmes d'une fonction $f : X \mapsto \mathbb{R}$, où X est un ensemble quelconque appelé espace de recherche et f est appelée fonction d'adaptation ou fonction d'évaluation ou encore fonction fitness. La fonction agit comme une «boite noire»pour l'algorithme génétique [[20],[21], [22], [23]]. Pour utiliser l'algorithme, on doit disposer des cinq éléments suivants :

i. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Le codages binaires ont été très utilisés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles.

ii. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

iii. Une fonction à optimiser. Celle-ci retourne une valeur appelée fitness ou fonction d'évaluation de l'individu.

iv. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.

v. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure 2.1 :

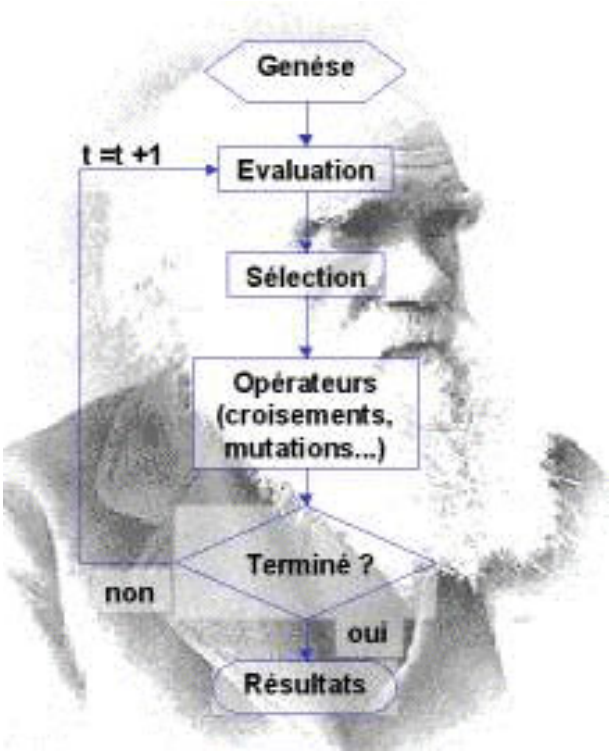


FIGURE 2.1 – Algorithme génétique

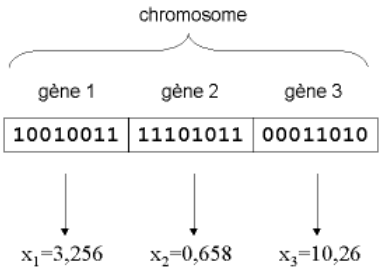


FIGURE 2.2 – Le codage réel

2.3 Description détaillée

2.3.1 Codage des données

Le premier pas dans l'implémentation des algorithmes génétiques est de créer une population d'individus initiaux. En effet, les algorithmes génétiques agissent sur une population d'individus, et non pas sur un individu isolé. Par analogie avec la biologie, chaque individu de la population est codé par un chromosome ou génotype. Une population est donc un ensemble de chromosomes. Chaque chromosome code un point de l'espace de recherche. L'efficacité de l'algorithme génétique va donc dépendre du choix du codage d'un chromosome.

- **Le Codage binaire :**
  Historiquement le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'état. Ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus. Cependant, ce type de codage n'est pas toujours bon comme le montrent les deux exemples suivants :
  1- Deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un codage de Gray.
  2- Pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais. Généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, l'ordre des variables ayant une importance dans la structure du chromosome alors qu'il n'en a pas forcément dans la structure du problème.
- **Le Codage réel :** cela peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle (voir la figure 2.2).
- **Codage de Gray :** dans le cas d'un codage binaire on utilise souvent la "distance de Hamming" comme mesure de la dissimilarité entre deux éléments de population, cette

mesure compte les différences de bits de même rang de ces deux séquences. Et c'est là que le codage binaire commence à montrer ses limites. En effet, deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "codage de Gray" : le codage de Gray est un codage qui a comme propriété que entre un élément n et un élément n + 1, donc voisin dans l'espace de recherche, un seul bit diffère.

- **Codage sous forme d'arbre**
  Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes où les solutions n'ont pas une taille finie. En principe, des arbres de taille quelconque peuvent être formés par le biais d'enjambements et de mutations.
  Le problème de ce type de codage est que les arbres résultants sont souvent difficiles à analyser et que l'on peut se retrouver avec des arbres « solutions » dont la taille sera importante alors qu'il existe des solutions plus simples et plus structurées à côté desquelles sera passé l'algorithme. De plus, les performances de ce type de codage par rapport à des codages en chaînes n'ont pas encore été comparées ou très peu. En effet, ce type d'expérience ne fait que commencer et les informations sont trop faibles pour se prononcer.

Finalement, le choix du type de codage ne peut pas être effectué de manière sûre dans l'état actuel des connaissances. Selon les chercheurs dans ce domaine, la méthode actuelle à appliquer dans le choix du codage consiste à choisir celui qui semble le plus naturel en fonction du problème à traiter et développer ensuite l'algorithme de traitement [[20],[21], [22]].

2.3.2 Génération de la population initiale

Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

2.3.3 Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. La figure (2.3) décrit le croisement en un point :

Il est tout à fait possible de faire des croisements aléatoires. Toutefois, une solution largement utilisée est d'effectuer des croisements multi-points (figure 2.4).

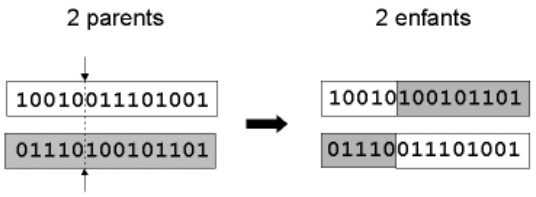


FIGURE 2.3 – Croisement en un point d'un individu de 14 bits

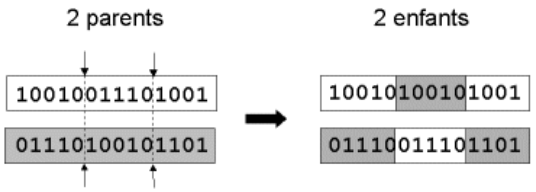


FIGURE 2.4 – Croisement en deux point d'un individu de 14 bits

Dans le schéma de croisement uniforme (figure 2.5), les bits de la chaîne sont comparées entre les deux parents. Les bits sont échangés avec une probabilité fixe, en général 0.5.

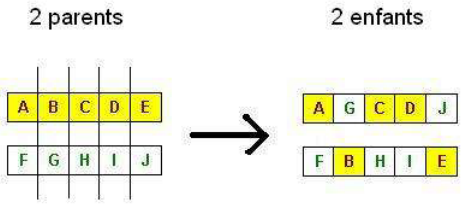


FIGURE 2.5 – Croisement uniforme

Sa probabilité d'apparition est un paramètre de l'algorithme génétique et dépend du problème et de la technique de recombinaison. La probabilité d'un croisement est alors comprise entre 0 et 1 strictement.

2.3.4 Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi, en toute rigueur, l'algorithme génétique peut converger sans croisement. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique. Pour les problèmes discrets, l'opérateur

de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir la figure 2.6).

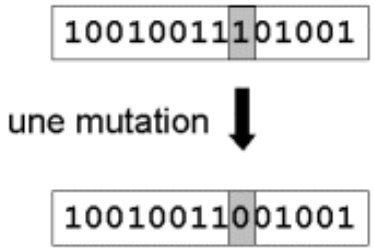


FIGURE 2.6 – La mutation

2.3.5 Principes de sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèse particulière sur la régularité de la fonction objectif. L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent.

Le peu d'hypothèses requises permet de traiter des problèmes très complexes. La fonction à optimiser peut ainsi être le résultat d'une simulation. La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent.

2.3.5.1 La roulette

La sélection des individus par le système de la roulette s'inspire des roues de loterie. A chacun des individus de la population est associé un secteur d'une roue. L'angle du secteur étant proportionnel à la qualité de l'individu qu'il représente. Vous tournez la roue et vous obtenez un individu. Les tirages des individus sont ainsi pondérés par leur qualité. Ainsi, les meilleurs individus ont plus de chance d'être croisés et de participer à l'amélioration de notre population (voir la figure 2.7).

2.3.5.2 La sélection par rang

La sélection par rang est une variante du système de roulette. Il s'agit également d'implémenter une roulette, mais cette fois ci les secteurs de la roue ne sont plus proportionnels à la qualité des individus, mais à leur rang dans la population triée en fonction de la qualité des individus.

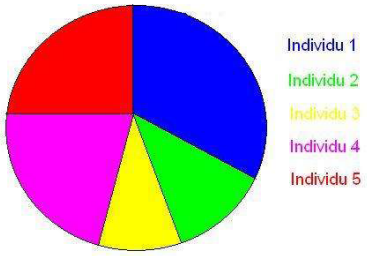


FIGURE 2.7 – Schéma d'une roulette

D'une manière plus parlante, il faut trier la population en fonction de la qualité des individus puis leur attribuer à chacun un rang. Les individus de moins bonne qualité obtiennent un rang faible (à partir de 1). Et ainsi, en itérant sur chaque individu, on finit par attribuer le rang N au meilleur individu (où N est la taille de la population). La suite de la méthode consiste uniquement en l'implémentation d'une roulette basée sur les rangs des individus. L'angle de chaque secteur de la roue sera proportionnel au rang de l'individu qu'il représente.

2.3.5.3 La sélection par tournoi

Le principe de la sélection par tournoi augmente les chances pour les individus de piètre qualité de participer à l'amélioration de la population. Le principe est très rapide à implémenter. Un tournoi consiste en une rencontre entre plusieurs individus pris au hasard dans la population. Le vainqueur du tournoi est l'individu de meilleure qualité. Nous pouvons choisir de ne conserver que le vainqueur comme nous pouvons choisir de conserver les 2 meilleurs individus ou les 3 meilleurs. A nous de voir, selon que nous souhaitons créer beaucoup de tournois, ou bien créer des tournois avec beaucoup de participants ou bien mettre en avant ceux qui gagnent les tournois haut la main. Nous pouvons faire participer un même individu à plusieurs tournois. Une fois de plus, nous sommes totalement libres quant à la manière d'implémenter cette technique de sélection.

2.3.5.4 L'élitisme

Cette méthode de sélection permet de mettre en avant les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population. Cette méthode a l'avantage de permettre une convergence (plus) rapide des solutions, mais au détriment de la diversité des individus. On prend en effet le risque d'écarter des individus de piètre qualité, mais qui auraient pu apporter de quoi créer de très bonnes solutions dans les générations suivantes. Cette méthode est extrêmement sensible à la présence d'optimas locaux, c'est à dire à la présence de solutions paraissant optimales tant que l'on ne s'en éloigne pas trop, le véritable optimum pouvant alors être situé dans

une toute autre partie du domaine de recherche. Pour s'assurer que les meilleurs individus feront effectivement partie de la prochaine génération, une autre possibilité relevant aussi du domaine de l'élitisme est tout simplement de les sauvegarder pour pouvoir les rajouter à coup sûr dans la population suivante.

2.3.6 L'insertion des nouveaux individus dans la population

Une fois que nous avons créé de nouveaux individus que ce soit par croisement ou par mutation, il nous faut sélectionner ceux qui vont continuer à participer à l'amélioration de notre population. Une fois encore, libre au programmeur de choisir ceux qu'il souhaite conserver. Il est possible de refaire une étape d'évaluation des individus nouvellement créés. De même qu'il est possible de conserver tous les nouveaux individus en plus de notre population.

Il n'est toutefois pas recommandé de ne conserver que les nouveaux individus et d'oublier la population de travail. En effet, rien ne nous dit que les nouveaux individus sont meilleurs que les individus de départ.

Une méthode relativement efficace consiste à insérer les nouveaux individus dans la population, à trier cette population selon l'évaluation de ses membres, et à ne conserver que les N meilleurs individus. Le nombre d'individus N à conserver est à choisir avec soin. En prenant un N trop faible, la prochaine itération de l'algorithme se fera avec une population plus petite et elle deviendra de plus en plus petite au fil des générations ; elle pourrait même disparaître. En prenant un N de plus en plus grand, nous prenons le risque de voir exploser le temps de traitement puisque la population de chaque génération sera plus grande.

Une méthode efficace est de toujours garder la même taille de population d'une génération à l'autre. Ainsi, il est possible de dérouler l'algorithme sur un grand nombre de générations.

2.4 Convergence

Les premiers théorèmes de convergence stochastiques des algorithmes génétiques ont été établis par John Holland sur le codage par chaîne de bits avec la roulette de sélection, le slicing crossover et la mutation classique. Ils utilisent la théorie des schémas : un schéma H de longueur $l(H)$ est une séquence particulière de gènes de longueur $l(H)$ représentée par une chaîne de caractères issus de l'alphabet $\{0, 1, *\}$ (pour le codage binaire), où * représente 0 ou 1. On dit qu'un chromosome A est une instance d'un certain schéma si la substitution des * par 0 ou 1 peut fournir un chromosome identique : ainsi, le chromosome 1010 est une instance des schémas 10**, 1**0 ou encore 101* (parmi d'autres). L'ordre $o(H)$ d'un schéma est le nombre de symboles différents de * qu'il contient, et sa longueur fondamentale $d(H)$ est la distance séparant les positions des symboles différents de * les plus proches des extrémités : les trois schémas de l'exemple précédent ont donc des ordres respectifs de 2, 2 et 3 et des longueurs fondamentales respectives de 2, 4, 3.
Le résultat principal de ces travaux montre que le nombre de représentants d'un schéma dans

la population suit une progression géométrique au cours des générations, et que les schémas favorisés sont ceux qui ont une fitness supérieure à la moyenne des fitness et un ordre ainsi qu'une longueur fondamentale faibles. D'autres résultats issus de la théorie des schémas montrent de plus qu'un algorithme génétique muni des trois opérateurs classiques réalise un compromis « quasi optimal » entre l'exploration de l'espace de recherche et l'exploitation des solutions déjà visitées. En outre, le nombre de schémas manipulés pour une population de taille n est de l'ordre de n^3, propriété appelée le parallélisme implicite [20].
Cependant, la démonstration de l'efficacité des algorithmes génétiques repose sur l'hypothèse des building blocks, c'est-à-dire de l'existence de schémas de longueur fondamentale et d'ordre faibles qui, incorporés au chromosome d'un individu, tentent de faire accroître sa fitness. Néanmoins, ces hypothèses sont en pratique peu probables sur des problèmes difficiles traités par les algorithmes génétiques, à moins de fournir un effort intensif lors du codage des données. Hormis les nombreux raffinements sur la théorie des schémas, d'autres tentatives ont été conduites pour rendre compte des performances des algorithmes génétiques. Raphaël Cerf [24] notamment présente des résultats de convergence asymptotique en utilisant une modélisation par les chaînes de Markov et la théorie de Freidlin-Wentzell sur les perturbations aléatoires de systèmes dynamiques, pour un algorithme génétique muni des trois opérateurs classiques avec des gènes sur des ensembles finis (non binaires).

2.5 Avantages et inconvénients

Les algorithmes génétiques sont des outils efficaces pour une classe de problèmes très large. De plus, ils permettent de traiter des problèmes où la fonction à optimiser ne présente aucune propriété de continuité ou de dérivabilité, par exemple. Néanmoins, ils présentent, aussi, un certain nombre de limitations :

1. Ils sont moins efficaces qu'un algorithme déterministe spécifique (lorsqu'il en existe un).
2. Les nombreux paramètres qui les contrôlent sont délicats à régler (probabilités de croisement et de mutation notamment, ainsi que le codage des chromosomes qui peut faire varier radicalement la vitesse de convergence).
3. Afin de garantir la robustesse des algorithmes évolutifs, le calcul d'un très grand nombre de fitness (parfois de l'ordre de plusieurs centaines de milliers) est généralement nécessaire avant l'obtention d'une bonne solution. Ce nombre de calcul important peut s'avérer problématique lorsque le coût de calcul (ressources systèmes ou temporelles) de la fitness est important, lorsqu'on travaille en grande dimension sur des fonctions à complexité importante par exemple.
4. Ils peuvent éprouver des difficultés à gérer des contraintes nombreuses et complexes car la technique des pénalités intégrées à la fonction de coût utilise des contraintes a posteriori de manière passive.

2.6 Conclusion

Les algorithmes génétiques, introduits en 1975 par Jean Holland, sont relativement récents. De ce fait, les fondements théoriques sont en cours de développement.
Malheureusement, Les algorithmes génétiques seuls ne sont pas très efficaces dans la résolution d'un problème. Ils apportent cependant assez rapidement une solution acceptable. Néanmoins, il est possible de les améliorer assez efficacement en les combinant avec un algorithme déterministe.

Chapitre 3

Optimisation par recuit simulé

3.1 Introduction

Le recuit simulé est le résultat d'expériences réalisées par Metropolis et al. dans les années 50 pour simuler l'évolution de ce processus de recuit physique. En revanche, l'utilisation de recuit simulé pour la résolution des problèmes d'optimisation combinatoire est beaucoup plus récente et date des années 80. La structure de configuration compliquée d'un espace de recherche d'un problème difficile donné a conduit trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi à proposer en 1983 une nouvelle méthode itérative qui permet d'éviter les minimas locaux [25]. Un travail similaire à été développé en parallèle par V. Cerny en 1985 [26].
Dans ce chapitre, nous allons commencer par expliquer le principe de fonctionnement de recuit simulé en donnant ses paramètres, ses critères de convergence pour finir avec quelques avantages et inconvénients de l'algorithme.

3.2 Principes généraux

L'algorithme du Recuit Simulé (SA) permet de résoudre le problème de minimum local. En effet, contrairement à la méthode du Gradient, un nouveau trajet de coût supérieur à celui du trajet courant ne sera pas forcément rejeté, son acceptation sera déterminée aléatoirement en tenant compte de la différence entre les coûts ainsi que d'un autre facteur appelé *température* [[27], [28], [29], [30] et les références incluses].
Ce paramètre, la "température", sert à prendre en compte le fait que plus le processus d'optimisation est avancé, moins on est prêt à accepter une solution plus coûteuse, ou alors, elle ne doit pas être trop coûteuse. Par contre, au début, l'acceptation de solutions fortement coûteuses permettra de mieux explorer tout l'espace des solutions possibles et par là-même, d'accroître nos chances d'approcher le minimum global.

L'idée est d'effectuer un mouvement selon une distribution de probabilité qui dépend de la qualité des différents voisins :

- Les meilleurs voisins ont une probabilité plus élevée.
- Les moins bons ont une probabilité plus faible.

3.3 Paramètres de l'algorithme

Lors de l'exposé de l'algorithme de recuit, apparait un certain nombre de paramètres et de termes qu'il faut expliciter.

- **Température (T) :** la température varie au cours de la recherche : T est élevée au début, puis diminue et finit par tendre vers 0.
  - T élevée : tous les voisins ont à peu près la même probabilité d'être acceptés.
  - T faible : un mouvement qui dégrade la fonction de coût a une faible probabilité d'être choisi.
  - T=0 : aucune dégradation de la fonction de coût n'est acceptée.
- **Équilibre statistique :** détermine le moment où l'on va changer de température, c'est à dire le nombre d'itérations que l'on va faire à la même température.
- **Gel du système :** signale la fin du traitement et correspond au fait qu'aucune transformation n'est plus acceptable. On peut choisir différentes approches, celle retenue pour le moment consiste à fixer par avance le nombre de paliers de température.

3.4 Description de l'algorithme de recuit simulé

Le recuit simulé s'appuie sur :

- l'algorithme de Metropolis-Hastings, qui permet de décrire l'évolution d'un système thermodynamique.
- la statistique de Boltzmann pour la description des phénomènes physiques et quantiques.

Lorsque l'équilibre thermodynamique est atteint à une température donnée, la probabilité pour un système physique d'avoir une énergie donnée E, est proportionnelle au facteur de Boltzmann : $e^{\frac{-E}{k_B T}}$ où k_B désigne la constante de Boltzmann. La distribution des états d'énergie est la distribution de Boltzmann à la température considérée.
En outre, afin de simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température donnée on peut utiliser l'algorithme de Metropolis : en se basant sur une configuration donnée, le système est soumis à une modification élémentaire. Si cette transformation entraîne une diminution de la fonction objective (ou «énergie») du système, elle est admise. Au contraire, si elle provoque une augmentation ΔE de le fonction objectif, elle est acceptée aussi mais avec une probabilité de $e^{\frac{-E}{k_B T}}$ (En pratique, cette condition est réalisée de le la manière suivante : un nombre réel est tiré au hasard compris entre 0 et 1,

et la configuration causant une dégradation ΔE de la fonction objectif est acceptée, si le nombre aléatoire tiré est inférieur ou égal à $e^{\frac{-\Delta E}{T}}$).
En observant à plusieurs reprises cette règle d'acceptation de Metropolis, une séquence de configurations est générée, ce qui constitue une chaîne de Markov (dans le sens que chaque configuration dépend seulement de celle qui la précède immédiatement).

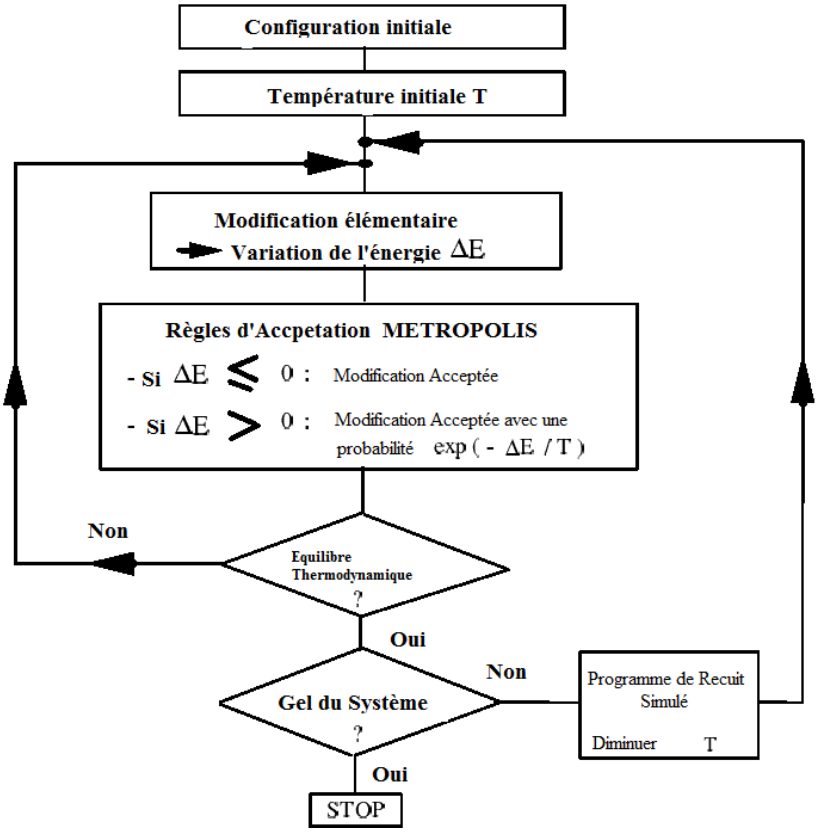


FIGURE 3.1 – Le schéma général de l'algorithme de recuit simulé

3.5 Convergence théorique de l'algorithme

Plusieurs recherches ont été menées sur la convergence de l'algorithme de recuit simulé. Certaines ont même essayé de développer un modèle général pour l'analyse des méthodes stochastiques pour l'optimisation globale. Le résultat principal de ces études théoriques est le suivant : sous certaines conditions, le recuit simulé converge sans doute vers un optimum global [31], dans le sens qu'on peut obtenir une solution proche de l'optimum.

Les preuves de convergence reposent sur le principe de construction de l'algorithme du recuit simulé :

- Sous réserve qu'elle respecte la condition de Doeblin [32], une chaîne de Markov converge vers une probabilité invariante unique, i.e. une probabilité asymptotiquement stable par application de la matrice de transition associée à la chaîne de Markov.
- L'algorithme de Metropolis [33] permet de construire la matrice de transition d'une chaîne de Markov de probabilité invariante donnée.

Ce résultat est, lui-même, significatif car il distingue le recuit simulé d'autres concurrents méta heuristique, dont la convergence n'est pas garantie. Il a également été démontré que la convergence est garantie à condition que la réversibilité est respectée ainsi que la connectivité (n'importe quel état du système peut être atteint à partir de tout autre état en effectuant un certain nombre fini de changements élémentaires) de l'espace de recherche. Cette formalisation présente deux avantages :

- Elle nous permet de baisser la température en passant d'un stade à un autre, ce qui améliore la vitesse de convergence de l'algorithme.
- Elle nous permet d'établir qu'une "bonne solution" (proche de l'optimum global) peut être obtenue par recuit simulé en un temps polynomial, pour certains problèmes NP-difficiles.

La vitesse de convergence de la méthode de recuit simulé dépend de deux facteurs : la configuration de l'espace de recherche et le programme de recuit simulé.

3.5.1 La configuration de l'espace de recherche

La configuration de l'espace joue un rôle fondamental dans la résolution des problèmes d'optimisation compliqués par le recuit simulé. Il possède une "topologie" générée par le concept de proximité entre deux configurations : la distance entre deux configurations représente le nombre minimum de changements élémentaires qu'on a besoin pour passer d'une configuration à une autre.

3.5.2 le schéma de recuit simulé

Deux approches sont possibles quant à la variation de la température :

1. Pour la première, on itère en gardant la température constante. Lorsque le système a atteint un équilibre thermodynamique (au bout d'un certain nombre de changements), on diminue la température du système. On parle alors de paliers de température.
2. La seconde approche fait baisser la température de façon continue. On peut alors imaginer toute sorte de loi de décroissance. La plus courante étant $Ti + 1 = X * Ti$ avec $X < 1$ (assez couramment $X = 0.99$).

Dans les 2 cas, si la température a atteint un seuil assez bas, fixé au préalable, ou que le système devient figé, l'algorithme s'arrête.

3.6 Avantages et inconvénients

Les méthodes de recuit simulé ont l'avantage d'être souples et rapidement implémentables lorsqu'on veut résoudre des problèmes d'optimisation combinatoire, le plus souvent de grande taille.
Les principaux inconvénients de recuit simulé résident dans le choix des nombreux paramètres, tels que la température initiale, la loi de décroissance de la température, les critères d'arrêt ou la longueur des paliers de température. Ces paramètres sont souvent choisis de manière empirique.

3.7 Conclusion

Le recuit simulé est une méthode d'optimisation importante historiquement. Grâce à son implémentation simple et ses propriétés de convergence intéressantes, elle trouve son application dans de nombreux domaines dans lesquels on a à résoudre des problèmes d'optimisation difficiles.

Chapitre 4

L'algorithme d'entropie croisée

4.1 Introduction

La méthode d'entropie croisée ou Cross Entropy (CE) est une approche Monte Carlo pour l'optimisation combinatoire et continue. Elle est attribuée à Reuven Rubinstein ([34], [35], [36] et les références incluses). Cette méthode trouve ses origines dans le domaine de simulations des événements rares où les petites probabilités doivent être estimées avec précision.
En effet, l'évaluation de telles probabilités est un problème difficile, crucial pour de nombreux systèmes stochastiques. D'ailleurs, il n'existe que quelques problèmes pour lesquels cette évaluation est possible de manière directe. Dans la plupart des cas, un recours à une méthode de simulation est donc nécessaire. Cependant, si la loi à évaluer est utilisée directement pour obtenir l'évènement rare, une simulation de Monte-Carlo simple devient rapidement inutilisable du fait du grand nombre de tirages à effectuer.
Classiquement, les techniques d'échantillonnage préférentiel (importance sampling) sont alors employées : il s'agit de favoriser l'occurrence de l'évènement rare en procédant à un changement de loi de tirage. En général, le paramétrage de ce changement de loi est difficile à mettre en oeuvre. La CE permet une évaluation simple itérative, basée sur la distance de Kullback-Leiber. De nombreuses applications ont montré que la CE est aussi efficace pour l'optimisation de problèmes combinatoires. Dans ce cadre, le problème déterministe à résoudre est translaté en un problème stochastique.
La solution de ce problème est alors approximée par une variable aléatoire.

4.2 Principes généraux

L'objet de la méthode CE est de "déformer" la variable aléatoire jusqu'à obtenir quelque chose de très "resserrée" autour de l'optimum du problème. La valeur optimale a alors une

forte probabilité d'être tirée.
La méthode CE implique un processus itératif où chaque itération peut être décomposée en deux phases :

- Génération aléatoire d'un échantillon d'information selon un mécanisme bien déterminé.
- Mise à jour de paramètres de la génération aléatoire. Cette phase implique la minimisation selon le principe d'entropie croisée ou la divergence de Kullback Leiber.

Considérons f une fonction de variable $\omega \in \Omega$ (Ω est un ensemble mesurable)et à variables réelles. L'objectif est d'optimiser la variable ω de manière à maximiser $f(\omega)$ jusqu'à ce qu'elle dépasse un seuil $\gamma \in \mathbb{R}$. Dans ce but, une famille de loi de probabilité est définie $(\pi(.;\lambda)|)$. La figure 4.1 décrit l'algorithme CE pour l'optimisation

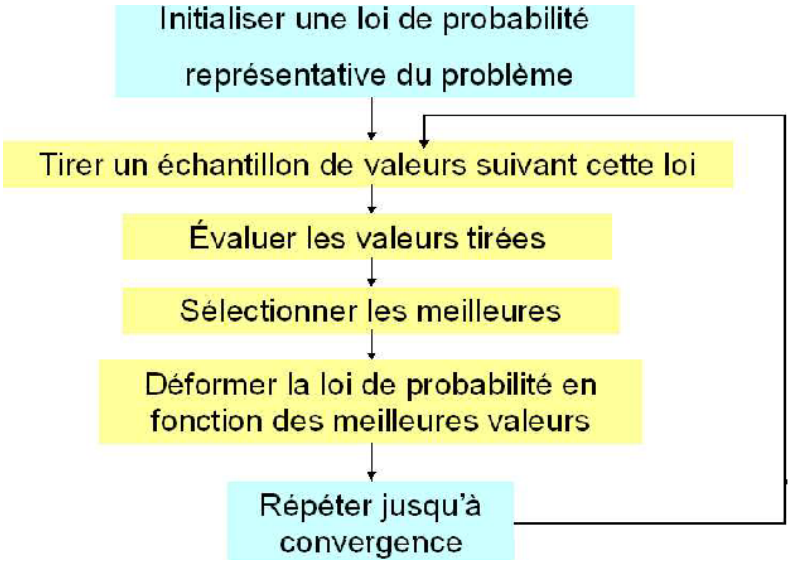


FIGURE 4.1 – L'algorithme d'entropie croisée

Algorithme 4.2.1
Soit un paramètre de sélection $\rho \in]0,1[$. L'algorithme d'entropie croisée s'énonce ainsi

1. *Choisir un vecteur des paramètres initial λ_0 ; poser $t = 1$.*
2. *Générer un échantillon de variables aléatoires $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ selon la loi $\pi(.;\lambda_{t-1})$.*
3. *Sélectionner les $\lfloor \rho_N \rfloor$ meilleurs échantillons selon la valeur de la fonction f. Soit $S_t \subset \mathbf{X}$ l'ensemble de ces échantillons.*
4. *Calculer λ_t par la maximisation suivante (distance de Kullback Leiber)*

$$\lambda_t \in arg\ max_\lambda \sum_{\omega \in S_t} \ln \pi(\omega;\lambda)$$

5. *Si l'algorithme a convergé alors stopper ; sinon, incrémenter t de 1 recommencer à l'étape 2.*

4.3 Paramètres de l'algorithme

Le choix de paramètres de l'algorithme CE est une étape cruciale lors de son implémentation. En effet, ces paramètres vont conditionner la convergence de la méthode.

- Paramètre de sélection ρ : il va caractériser la vitesse de convergence. En effet, la borne γ_t définie à chaque étape de l'algorithme est définie par le $(1-\rho)$-quantile de $f(x)$ pour la loi $\pi(.;\lambda_{t-1})$
  Une définition plus claire de γ_t serait : soient $(\omega_1 \cdots \omega_N)$ les échantillons générés par $\pi(.;\lambda_{t-1})$. Supposons ces échantillons ordonnés de manière à avoir $f(\omega_i) \leq f(\omega_{i+1})$ pour $1 \leq i \leq N$. Alors γ_t est définie par $\gamma_t = \min\left\{\gamma, f(\omega_{\lceil (1-\rho)N \rceil})\right\}$. Le paramètre ρ doit être choisi suffisamment grand.
- La famille de lois : elle servira pour choisir la loi d'échantillonnage $\pi(.;\lambda)$. Toutefois, les lois utilisées se déduisent naturellement du problème posé.
- Le critère d'arrêt : il existe plusieurs critères d'arrêt pour l'algorithme CE. On peut citer le nombre maximum d'itération ou bien le fait que γ_t ne change plus pendant un nombre déterminé successif d'itérations. La définition d'un bon critère d'arrêt nécessite une part d'expérimentation.

4.4 Convergence

La convergence de l'algorithme CE lors d'un nombre fini d'itérations pour un échantillon de variables aléatoires déterminé au début est détaillée dans [37]. Dans le même article, les auteurs ont montré que pour une petite valeur de ρ et une grande valeur de N, la convergence de l'algorithme CE vers l'optimum est exponentielle pour N échantillons de variables aléatoires.

4.5 Avantages et inconvénients

La méthode CE est une méthode d'optimisation stochastique basée sur le principe d'échantillonnage préférentiel. Elle fait partie des méthodes de simulation de Monte-Carlo. En tant que méthode de simulation, elle s'adapte à des données difficiles à formaliser. Cependant, le principal inconvénient qu'elle présente est le temps de calcul important.

4.6 Conclusion

Dans ce chapitre, nous avons expliqué brièvement le principe de la méthode CE. Cette méthode, qui trouve ses origines dans le domaine de simulations des événements rares, elle est aujourd'hui utilisée dans plusieurs domaines [[38], [39], [40], [41]]. La méthode CE peut être appliquée à tout problème d'optimisation combinatoire où les observations sont bruitées comme le problème du voyageur de commerce, le problème d'affectation quadratique, le problème d'alignement de séquences et le problème de la coupure maximale, tout comme des problèmes d'optimisation continue avec de nombreux extrema locaux.

Chapitre 5

Méthodes hybrides

5.1 Introduction

La résolution d'un problème d'optimisation consiste à explorer un espace de recherche afin de maximiser (ou minimiser) une fonction donnée. Pour être performante, une méthode de résolution doit associer judicieusement exploration et exploitation de l'espace de recherche. Or cette méthode est rarement aussi efficace pour exploiter que pour explorer l'espace de recherche. Cette difficulté dûe aux complexités (en taille ou en structure) relatives de l'espace de recherche et de la fonction à maximiser conduit à utiliser des méthodes de résolutions radicalement différentes. Une première approximation consiste à dire qu'une méthode déterministe est adaptée à un espace de recherche petit et complexe et qu'un espace de recherche grand nécessite plutôt une méthode de recherche stochastique (recuit simulé, algorithme génétique, ...). Une solution consiste à ajouter des mécanismes complémentaires dans une méthode de résolution donnée. Il peut être extrêmement bénéfique d'associer une méthode de recherche dont les caractéristiques d'exploration sont très élevées à une méthode de recherche dont le point fort est l'exploitation. D'où l'idée de méthodes hybrides.

Les méthodes hybrides permettent non seulement d'élargir le spectre d'application de certaines méthodes de résolution mais aussi d'augmenter leurs performances. Pour appliquer efficacement ces techniques, nous devons avoir une bonne visibilité vis à vis de points forts de chaque méthodes à part. Par exemple, les algorithmes génétiques sont très performants lorsqu'il s'agit d'explorer l'espace de recherche, mais ils s'avèrent ensuite incapables d'exploiter efficacement la zone vers laquelle la population converge. Il est alors bien plus intéressant (en terme de durée d'exécution et de qualité de solutions) de stopper l'algorithme génétique pour utiliser une autre méthode. L'hybridation peut, aussi, être utilisée pour résoudre simultanément différents aspects d'un même problème : cette méthode est souvent utilisée dans le domaine de la gestion de production [[42], [43]] où se posent simultanément différents problèmes tels que les affectations de machines, des personnels et des opérations aussi bien que la gestion de stocks ...

5.1.1 Choix des méthodes à hybrider

Il est possible d'hybrider toutes les techniques, y compris méthodes exactes et heuristiques [44]. Pratiquement, le souci de performances et les ressources informatiques limitent les possibilités d'hybridation. De ce fait, on doit être prudent vis à vis des techniques utilisées pour obtenir une bonne coopération entre les constituants et les méthodes hybrides.

5.1.2 Les techniques d'hybridation

L'hybridation peut prendre place dans un ou plusieurs composants d'une méthode de recherche. Elle peut également consister à assembler plusieurs méthodes d'hybridation pour former une seule méthode hybride. Les différentes techniques d'hybridation peuvent être réparties en trois catégories principales [45].

- Hybridation séquentielle ;
- Hybridation parallèle synchrone ;
- Hybridation parallèle asynchrone.

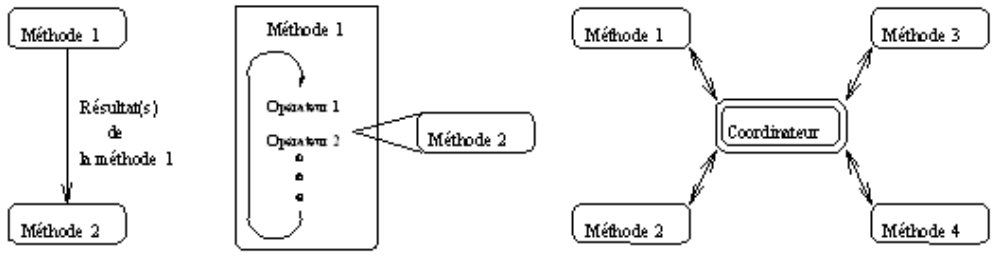


FIGURE 5.1 – Techniques d'hybridation.

L'hybridation *séquentielle* consiste à exécuter séquentiellement différentes méthodes de recherche de telle façon que le (ou les) résultat(s) d'une méthode serve(nt) de solution(s) initiale(s) à la suivante. Cette technique d'hybridation est la plus simple, elle ne nécessite pas de modification des méthodes de résolution utilisées : il suffit d'initialiser chaque méthode à partir de solutions pré-calculées.
L'hybridation *parallèle synchrone* est obtenue en incorporant une méthode de résolution dans une autre. C'est une technique plus fine que la précédente. En effet, il faut tenir compte des fortes interactions entre les méthodes dans ce type d'hybridation.
L'hybridation *parallèle asynchrone* consiste à faire évoluer en parallèle différentes méthodes de résolution. Cette co-évolution permet une bonne coopération des méthodes de résolution au travers d'un coordinateur chargé d'assurer le transfert d'informations entre les méthodes de résolution. Cette technique exige la modification de méthodes de résolution pour assurer la communication avec le coordinateur.

5.2 Les schémas d'hybridation proposés

Dans le cadre de nos travaux, nous avons appliqué la première et la deuxième technique d'hybridation.

5.2.1 Le schéma d'hybridation GA-DCA

Le but de l'hybridation dans ce schéma est de chercher des solutions admissibles puis rechercher la solution à coût minimal parmi ces dernières : c'est l'hybridation séquentielle. Dans le schéma GA-DCA, nous recourons à l'algorithme génétique pour une exploration rapide et globale d'un espace de recherche de taille importante, surtout qu'il est capable de fournir plusieurs solutions. Ensuite, nous donnons la main à une méthode déterministe locale qui est capable de donner une solution globale si elle commence par un bon point de départ (DCA).
Le schéma suivant décrit l'algorithme.

Algorithme 5.2.1

1. *AG − proc() //La procédure qui fait démarrer l'algorithme génétique pour l'exploration de l'espace de recherche.*
2. *Choisir le meilleur individu de la population selon la valeur de fitness.*
3. *DCA − proc() //La procédure DCA ayant pour point initial le meilleur individu choisi.*

5.2.2 Le schéma d'hybridation SA-DCA

Nous avons recouru à ce schéma SA-DCA afin d'exploiter simultanément les avantages de deux techniques (la méthode de recuit simulé (SA) avec l'approche (DCA)) à savoir : exploration maximale de l'espace de recherche en évitant les minimas locaux et les performances intéressantes de DCA si on commence par une bonne solution initiale.
Dans cette hybridation parallèle synchrone, nous appliquons DCA à chaque solution candidate acceptée par SA. Le schéma suivant décrit l'algorithme implémenté.
On désigne par :

- ICMAx : le nombre maximal des itérations qu'on peut atteindre.
- neighbour() : la procédure chargée de trouver le prochain candidat dans l'espace de recherche.
- Cost : la procédure chargée de calculer la fonction d'évaluation.

Algorithme 5.2.2

1. *Soit T0 la température de départ.*
2. *Soit IC = 0 (le compteur d'itérations).*

3. *Générer aléatoirement une solution initiale courante* $\hat{f}_{curr}$.
4. ***Tant que*** $(IC < ICMax)$
   $\{$

   Répéter N fois
   $\{$
   a- Générer $\hat{f}_{new} = neighbour\left(\hat{f}_{curr}\right)$.
   $\{$
   b- Calculer la différence entre $\hat{f}_{new}$ *et* $\hat{f}_{curr}$,
   $\Delta_{cost} = cost\left(\hat{f}_{new}\right) - cost\left(\hat{f}_{curr}\right)$

   c- Si $(\Delta_{cost} < 0)$ *alors*
   $\{$
   c-1 Accepter la solution candidate, $\hat{f}_{curr} = \hat{f}_{new}$.
   c-2 Appliquer la procédure DCA pour améliorer le résultat.
   $\}$
   d- Sinon
   générer une valeur u *aléatoire dans* $(0, 1)$.
   Si $\left(\exp^{-\frac{\Delta_{cost}}{T}} > u\right)$
   $\{$
   d-1 Accepter,
   d-2 Appliquer la procédure DCA procédure pour améliorer le résultat.

   $\}$
   Sinon rejeter la solution.
   $\}$

5. $T = T * \alpha$ *(The geometrical law of decrease).*
6. $IC = IC + 1$.
   $\}$

7. *Considérer la valeur courante de* $\hat{f}$ *comme la 'solution' finale.*

5.2.3 Le schéma d'hybridation CE-DCA

Afin de pallier au problème de temps de calcul important de la méthode CE, nous recourons à une procédure DCA appliquée à l'ensemble de solutions candidates les mieux classées

selon un modèle d'hybridation parallèle synchrone. Le schéma suivant décrit l'algorithme CE-DCA :

Algorithme 5.2.3

1. *Initialisation des paramètres*
2. *Tant que l'on n'a pas convergé*
   {
3. *Génèrer N échantillons à partir de la distribution*
4. *Calculer le score de chaque échantillon*
5. *Classer les échantillons selon le score*
6. *Appliquer DCA sur les meilleurs échantillons*
7. *Mettre à jour des parcmètres de la distribution*
   }

5.3 Conclusion

Une hybridation idéale engendre une méthode hybride qui cumule les bonnes propriétés de ses constituants tout en inhibant leurs points faibles. Dans les prochains chapitres, nous définissons les problèmes pour lesquels nous allons évaluer différentes méthodes hybrides.

Deuxième partie

Modélisation et optimisation pour la résolution de certains problèmes en cryptologie

Chapitre 6

Introduction à la cryptologie

Introduction

A coté du terme ancien de cryptographie, limité à l'origine au chiffrement des messages, on a introduit récemment, de manière à préciser la terminologie et rendre compte de l'évolution récente du domaine, la dénomination générale de cryptologie pour désigner la partie de la sécurité des systèmes d'information qui s'occupe d'assurer, ou au contraire de compromettre, les grandes fonctions de sécurité (en particulier la confidentialité, l'intégrité des données, l'authentification et la non-répudiation). Ainsi la cryptologie a un double aspect : d'une part la cryptographie qui consiste à construire des outils propres à assurer des fonctionnalités de sécurité et d'autre part la cryptanalyse qui consiste à s'attaquer aux outils en question pour en trouver les faiblesses.
Dans ce chapitre, On va commencer par définir la cryptologie dans ses deux aspects. Ensuite, on va décrire ses techniques pour finir avec une brève présentation de notre contribution.

6.1 Cryptographie : définition et algorithmes

6.1.1 Définition

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique.
Un message est appelé texte en clair. Le processus de transformation d'un message de telle manière à le rendre incompréhensible est appelé chiffrement. Le résultat du processus de chiffrement est appelé texte chiffré (ou cryptogramme). Le processus de reconstruction du texte en clair à partir du texte chiffré est appelé déchiffrement (ou décryptage)[46]. Ces différents processus sont illustrés par la figure (6.1) :

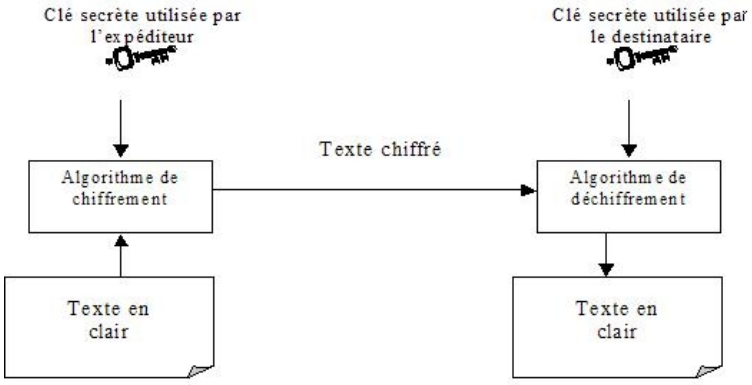


FIGURE 6.1 – Chiffrement et déchiffrement

Jusqu'en 1976, la seule méthode connue comme pour échanger de l'information était la cryptographie à clé secrète ; pour que deux personnes puissent échanger un message chiffré, elles devraient partager un même secret connu d'elles seules. En 1976, est apparue une nouvelle forme de chiffrement dite à clé publique où les protagonistes ne sont plus tenus de partager une même clé.

6.1.2 Techniques cryptographiques

6.1.2.1 chiffrement classique

La cryptographie classique concerne la période précédant les ordinateurs. Elle traite des systèmes reposant sur les lettres et les caractères d'une langue naturelle (allemand, anglais, français, etc...). Les principaux outils utilisés remplacent des caractères par d'autres et les transposent dans des ordres différents. Les meilleurs systèmes (de cette classe d'algorithmes) répètent ces deux opérations de base plusieurs fois. Cela suppose que les procédures (de chiffrement ou déchiffrement) soient gardées secrètes ; sans cela, le système est complètement inefficace. On appelle généralement cette classe de méthodes : le chiffrement à usage restreint.

La substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. Il existe 4 types de cryptage par substitution : simple, homophonique, polyalphabétique et polygrammique.

La substitution simple : chaque caractère du message en clair et provenant d'un alphabet noté A, est remplacé par le caractère correspondant appartenant à un alphabet de substitution noté S. L'alphabet S est un réarrangement de l'ordre lexicographique des caractères de A. Soit A un alphabet à n caractères : $\{a_0, a_1, a_3, \cdots, a_{n-1}\}$; C sera représenté

par : $\{f(a_0), f(a_1), f(a_3), \cdots, f(a_{n-1})\}$ où la transformation $f : A \longrightarrow S$ remplace chaque caractère de A par le caractère correspondant de S. La clé est déterminée ici par l'alphabet C ou la fonction f.
Pour le chiffrement, le message $M = m_1 m_2 m_3 \cdots$ est transformé en
$E_k(M) = f(m_1) f(m_2) f(m_3) \cdots$

Substitution homophonique : même principe que le cryptage précèdent si ce n'est que la transformation n'est plus bijective. Ainsi à une lettre peut correspondre plusieurs autres. Ainsi le décryptage se fait en suivant le bon sens de la phrase ou du mot.

Substitution Polyalphabétique : le principe consiste à remplacer chaque lettre du message en clair par une nouvelle lettre prise dans un ou plusieurs alphabets aléatoires associés. Par exemple, on pourra utiliser n substitutions monoalphabétiques ; celle qui est utilisée dépend de la position du caractère à chiffrer dans le texte en clair. On choisit une clé qui sert d'entrée dans la grille polyalphabétique incluant autant de symboles qu'il y a de lettres différentes à chiffrer. Chaque caractère de la clé désigne une lettre particulière dans la grille de codage. Pour coder un caractère, on doit lire le caractère correspondant du texte en clair en utilisant la grille polyalphabétique et le mot clé associé dans l'ordre séquentiel (on répète la clé si la longueur de celle-ci est inférieure à celle du texte de départ). L'exemple le plus célèbre est l'algorithme de VIGENERE et de BEAUFORT. L'illustration la plus simple qui corresponde à ce principe est l'utilisation d'une fonction à base de ou exclusif (XOR).

Substitution par polygrammes : les caractères du texte en clair sont chiffrés par blocs. Par exemple, " ABA " peut être chiffré par " RTQ " tandis que " ABB " est chiffré par " SLL ". Les exemples les plus célèbres sont les algorithmes de PLAYFAIR et de HILL inventés en 1854 et utilisés pendant la première guerre mondiale par les anglais.

La transposition

Avec le principe de la transposition toutes les lettres du message sont présentes, mais dans un ordre différent. Il utilise le concept mathématique des permutations. Pour déchiffrer le texte chiffré, il suffit d'écrire verticalement celui-ci sur un morceau de papier quadrillé de la même largeur et de lire horizontalement le texte clair. La cryptanalyse de ces chiffres est présentée dans [47]. Plusieurs types différents de transpositions existent.

Transposition simple par colonnes : on écrit le message horizontalement dans une matrice prédéfinie, et on trouve le texte à chiffrer en lisant la grille verticalement. Le destinataire légal pour décrypter le message réalise le procédé inverse.

Transposition complexe par colonnes : un mot clé secret (avec uniquement des caractères différents) est utilisé pour dériver une séquence de chiffres commençant de 1 et finissant au nombre de lettres composant le mot clé. Cette séquence est obtenue en numérotant les lettres du mot clé en partant de la gauche vers la droite et en donnant l'ordre d'apparition dans l'alphabet. Une fois la séquence de transposition obtenue, on chiffre en écrivant d'abord le message par lignes dans un rectangle, puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence.

Transposition par carré polybique : un mot clé secret est utilisé pour construire un alphabet dans un tableau. Les coordonnées des lignes et des colonnes correspondant aux lettres du texte à chiffrer sont utilisés pour transcrire le message en chiffres. Avec ce procédé chaque lettre du texte en clair est représenté par deux chiffres écrits verticalement. Ces deux coordonnées sont ensuite transposées en les recombinant par deux sur la ligne ainsi obtenue.

6.1.2.2 Chiffrement à clé publique

Le principe de chiffrement asymétrique (appelé aussi chiffrement à clés publiques) est apparu en 1976, avec la publication d'un ouvrage sur la cryptographie par Whitfield Diffie et Martin Hellman [[48], [49]]. C'est le début de l'ère de la cryptographie à clé publique et du règne du système RSA développé en 1977 par Rivest, Shamir et Adleman [50]. Aujourd'hui, il existe divers systèmes à clé publique.

Dans un cryptosystème asymétrique (ou cryptosystème à clés publiques), les clés existent par paires (le terme de bi-clés est généralement employé) :
- Une clé publique pour le chiffrement ;
- Une clé secrète pour le déchiffrement.

Ainsi, dans un système de chiffrement à clé publique, les utilisateurs choisissent une clé aléatoire qu'ils sont seuls à connaître (il s'agit de la clé privée). A partir de cette clé, ils déduisent chacun automatiquement un algorithme (il s'agit de la clé publique). Les utilisateurs s'échangent cette clé publique au travers d'un canal non sécurisé.

Lorsqu'un utilisateur désire envoyer un message à un autre utilisateur, il lui suffit de chiffrer le message à envoyer au moyen de la clé publique du destinataire (qu'il trouvera par exemple dans un serveur de clés tel qu'un annuaire LDAP). Ce dernier sera en mesure de déchiffrer le message à l'aide de sa clé privée (qu'il est seul à connaître) (voir 6.2).

Le problème consistant à se communiquer la clé de déchiffrement n'existe plus, dans la mesure où les clés publiques peuvent être envoyées librement. Le chiffrement par clés publiques permet donc à des personnes d'échanger des messages chiffrés sans pour autant posséder de secret en commun.

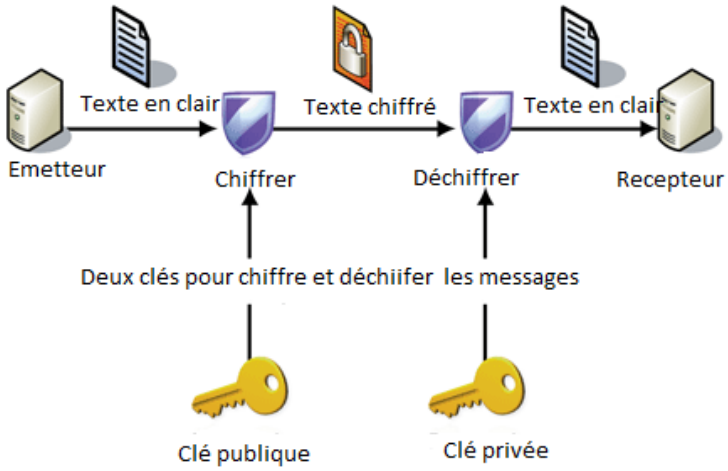


FIGURE 6.2 – chiffrement à clé publique

En contrepartie, tout le challenge consiste à s'assurer que la clé publique que l'on récupère est bien celle de la personne à qui l'on souhaite faire parvenir l'information chiffrée.
Depuis 1967, de nombreux algorithmes de cryptographie à clé publique ont été proposés. Nombre d'entre eux ne sont pas sûrs. Soit ils ont une clef exagérément grande, soit le texte chiffré est nettement plus long que le texte en clair.
Seuls quelques algorithmes sont à la fois sûrs et pratiques. Ces algorithmes sont en général basés sur des problèmes difficiles.

Exemple 6.1.1
Algorithme de chiffrement basé sur le problème de sac à dos
Cet algorithme est Considéré parmi les premiers cryptosystèmes à clef publique. Merkle et Hellman [51] ont proposé d'utiliser un sac à dos croissant comme clé privée, et de le camoufler sous un sac à dos général pour en faire une clé publique. Le premier schéma a été facilement décodé. Une première attaque a été réalisée par Brickell in 1983 [52]. Une année plus tard, Schamir a développé un algorithme polynomial[53] pour attaquer l'algorithme de chiffrement basique. Merkle et Hellman ont proposé une autre version de l'algorithme en utilisant le "trapdoor" sac à dos [51] qui a été attaqué par Richard Spillman en utilisant un algorithme génétique [54].

6.1.2.3 Chiffrement à clé secrète

Le chiffrement symétrique (aussi appelé chiffrement à clé privée ou chiffrement à clé secrète) consiste à utiliser la même clé pour le chiffrement et le déchiffrement (voir 6.3).
Le chiffrement à clé secrète a des origines lointaines, et a toujours été associé à des applications militaires.

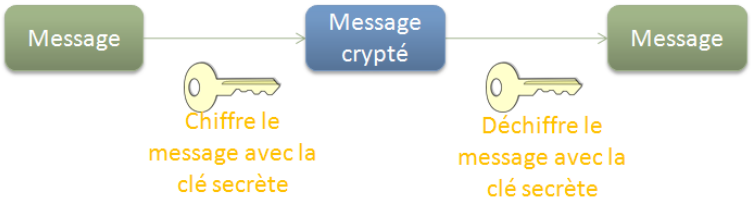


FIGURE 6.3 – chiffrement à clé secrète

Toutefois, dans les années 40, Claude Shannon démontra que pour être totalement sûr, les systèmes à clefs privées doivent utiliser des clefs d'une longueur au moins égale à celle du message à chiffrer. De plus le chiffrement symétrique impose d'avoir un canal sécurisé pour l'échange de la clé, ce qui dégrade sérieusement l'intérêt d'un tel système de chiffrement.

Le principal inconvénient d'un cryptosystème à clefs secrètes provient de l'échange des clés. En effet, le chiffrement symétrique repose sur l'échange d'un secret (les clés). Ainsi, se pose le problème de la distribution des clés.

D'autre part, un utilisateur souhaitant communiquer avec plusieurs personnes en assurant des niveaux de confidentialité distincts doit utiliser autant de clés privées qu'il a d'interlocuteurs. Pour un groupe de N personnes utilisant un cryptosystème à clés secrètes, il est nécessaire de distribuer un nombre de clés égal à $N * (N-1)/2$.

Ainsi, dans les années 20, Gilbert Vernam et Joseph Mauborgne mirent au point la méthode du One Time Pad (méthode du masque jetable, parfois appelé « One Time Password » et noté OTP), basée sur une clé privée, générée aléatoirement, utilisée une et une seule fois, puis détruite.

6.1.3 Signature numérique

6.1.3.1 Description

Un algorithme de signature numérique a plusieurs volets. Tout d'abord un condensé du message est calculé avec une fonction de hachage. Ainsi, le message M est transformé en un message $m = h(M)$ en général plus court, de longueur fixée, grâce à une fonction publique de hachage h. Pour la partie signature proprement dite, chaque utilisateur A dispose d'une clé publique e_A et d'une clé privée d_A. Le système définit une fonction de signature S qui à une clé privée d_A et à un message condensé m fait correspondre $s = S(d_A, m)$, appelé appendice de la signature de M par l'utilisateur A. La démarche est donc la suivante : l'utilisateur A, qui veut signer un message M, commence par en faire un condensé $m = h(M)$(la fonction h est publique et utilisable par tous), puis il calcule, grâce à sa clé privée, l'appendice $s = S(d_A, m)$. Il peut alors transmettre son message signé (M, s). Le système définit également une fonction

de vérification V qui à une clé publique e_A, et à un message signé (M, s) fait correspondre $V(e_A, M, s)$ valant "vrai" si (M, s) est signé par A et "faux" sinon.

6.1.3.2 Fonction de hachage

Une fonction de hachage (parfois appelée fonction de condensation) est une fonction permettant d'obtenir un condensé (appelé aussi condensat ou en anglais message digest) d'un texte, c'est-à-dire une suite de caractères assez courte représentant le texte qu'il condense. La fonction de hachage doit être telle qu'elle associe un et un seul condensat à un texte en clair (cela signifie que la moindre modification du document entraîne la modification de son condensat). D'autre part, il doit s'agir d'une fonction à sens unique (one-way function) afin qu'il soit impossible de retrouver le message original à partir du condensé. S'il existe un moyen de retrouver le message en clair à partir du condensé, la fonction de hachage est dite « à brèche secrète » (voir 6.4).

FIGURE 6.4 – Fonction de hachage

Ainsi, le condensé représente en quelque sorte l'empreinte digitale (en anglais finger print) du document.

Les algorithmes de hachage les plus utilisés actuellement sont :

- MD5 (MD signifiant Message Digest). Développé par Rivest en 1991, MD5 crée une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire en le traitant par blocs de 512 bits. Il est courant de voir des documents en téléchargement sur Internet accompagnés d'un fichier MD5, il s'agit du condensé du document permettant de vérifier l'intégrité de ce dernier)
- SHA (pour Secure Hash Algorithm, pouvant être traduit par Algorithme de hachage sécurisé) crée des empreintes d'une longueur de 160 bits
  SHA-1 est une version améliorée de SHA datant de 1994 et produisant une empreinte de 160 bits à partir d'un message d'une longueur maximale de 264 bits en le traitant par blocs de 512 bits.

6.1.4 Authentification : preuve à divulgation nulle

Une preuve à divulgation nulle de connaissance est un concept utilisé en cryptologie dans le cadre de l'authentification et de l'identification [[55], [56], [57],[58]]. Cette expression désigne un protocole sécurisé dans lequel une entité nommée « fournisseur de preuve », prouve mathématiquement à une autre entité, le « vérificateur », qu'une proposition est vraie sans toutefois révéler une autre information que la véracité de la proposition.

En pratique, ce schéma se présente souvent sous la forme d'un protocole de type « stimulation/réponse » (challenge-response). Le vérificateur et le fournisseur de preuve s'échangent des informations et le vérificateur contrôle si la réponse finale est positive ou négative.

6.1.4.1 Propriétés

Trois propriétés doivent être satisfaites :

- consistance (completeness) : si le fournisseur de preuve et le vérificateur suivent le protocole alors le vérificateur doit toujours accepter la preuve.
- solidité (soundness) : si la proposition est fausse, aucun fournisseur de preuve malicieux ne peut convaincre un vérificateur « honnête » que la proposition est vraie et ceci avec une forte probabilité.
- aucun apport d'information (zero knowledge) : le vérificateur n'apprend de la part du fournisseur de preuve rien de plus que la véracité de la proposition, il n'obtient aucune information qu'il ne connaissait déjà sans l'apport du fournisseur de preuve. Si le vérificateur ne suit pas la procédure, cette définition reste valable aussi longtemps que le fournisseur de preuve suit la procédure.

Les deux premières propriétés sont les mêmes qui servent à définir un système de preuve interactive, qui est un concept plus général. C'est la troisième propriété qui fait le zero knowledge.

6.1.4.2 Principes généraux

Chaque individu est le détenteur d'un secret s qui satisfait un prédicat $P(I, s)$ pour une instance I d'un problème difficile (le problème de la factorisation par exemple). Le vérificateur pose une série de questions au prouver. Si ce dernier connaît la quantité s, il peut répondre correctement à toutes les questions. Dans le cas contraire, la probabilité pour qu'un intrus réponde correctement vaut q $(0 < q < 1)$ cette valeur dépend essentiellement du nombre de questions posées). En répétant l protocole r fois, la probabilité de succès d'un fraudeur est de q^r. La valeur de r permet donc de fixer un seuil de sécurité au delà duquel le vérificateur s'estimera convaincu de l'identité de son interlocuteur.

6.1.4.3 Schéma basé sur Permuted Kernel Problem (PKP)

C'est un protocole proposé par Shamir [59].

Données :

- une matrice $A\,(m \times n)$,
- un nombre premier P,
- un vecteur V à n éléments.

Trouver :

- une permutation π telle que $V_\pi \in K\,(A)$,
- $K\,(A)$: ensemble de vecteurs w à n éléments tels que $Aw \equiv 0\,(modP)$,
- 0 : Vecteurs à m zéro éléments.

Pour utiliser le PKP dans les protocoles d'identification, les utilisateurs se mettent d'accord sur une matrice M et un nombre premier P. Ensuite, chaque utilisateur choisit une permutation π (qui va jouer le rôle de la clé secrète) et un vecteur V tel que $V_\pi \in K\,(A)$. Les utilisateurs sont appelés, maintenant, à dévoiler leurs identités en prouvant qu'ils connaissent la permutation secrète π et en utilisant la preuve à divulgation nulle de connaissance.

6.1.4.4 Schéma basé sur Permuted Perceptron Problem (PP)

C'est une technique proposée par Pointcheval [13] en se basant sur le problème NP-complet Perceptron Problem (PP) qui doit son existence au fameux problème perceptron dans les réseaux de neurones. La technique d'identification ainsi proposée est basée sur des opérations mathématiques simples (addition et soustraction).

Perceptron Problem

Données :

une matrice $A\,(m, n)$ dont tous les éléments sont dans l'ensemble $\{-1, 1\}$.

Trouver :

un vecteur dans $\{-1, 1\}$ de taille n tel que $(AV)_i \geq 0\ \forall i \in \{1 \cdots m\}$.

Permuted Perceptron Problem

Données :

une matrice $A\,(m, n)$ dont tous les éléments sont dans l'ensemble $\{-1, 1\}$, un tableau S de taille m d'entiers positifs.

Trouver :

un vecteur V de taille n tel que $\{\{(AV)_i\,|i = \{1 \cdots m\}\}\} \equiv S$.

Il est clair qu'une solution pour PPP est aussi une solution pour PP.

6.1.4.5 Schéma d'identification de Feige-Fiat-Shamir

Ce système, explicité dans [60], constitue une preuve à divulgation nulle. Il utilise la difficulté d'extraire une racine carrée dans $\mathbb{Z}/n\mathbb{Z}$, où n est un entier produit de deux grands

nombres premiers ($n = pq$) quand on ne connaît pas la factorisation de n. On dispose donc d'un groupe multiplicatif $G = (\mathbb{Z}/n\mathbb{Z})^*$ où $n = pq$ (p et q ont au moins 512 bits chacun). L'interlocuteur P dispose d'une clé privée a où a est un entier premier avec n tel que $1 \leq a \leq n-1$. La clé publique de P est $v = a^2 \ mod \ n$. Remarquons que v est aussi premier avec n. Le prouveur P va prouver au vérificateur V qu'il connaît une racine carré de v modulo n.
Si P connaît effectivement la clé privée a et effectue honnêtement l'échange, alors la vérification faite par V est concluante : en effet, si $b = 0$, V calcule $r^2 \ mod \ n$ et trouve bien le x envoyé par P précédemment ; si $b = 1$ alors V calcule :

$$\left(y^2 \ \times v^{-1}\right) \ mod \ n \ = \ \left(r^2 \times a^2 \times v^{-1}\right) mod \ n \ = r^2 \ mod \ n \ = \ x,$$

donc la vérification est concluante. Si maintenant P ne connaît pas la clé privée et essai de tricher :

- dans la perspective où V enverrait le bit 0, il doit choisir un r et envoyer son carré x.
- dans la perspective où V enverrait le bit 1, il doit engendrer un couple (x, y) tel que $x = \ (y^2 \times v^{-1}) \ mod \ n$ Sans connaître par avance la valeur du bit b, un intrus ne peut pas satisfaire simultanément les deux contraintes précédentes, sauf s'il connaît a. Il a donc une chance sur deux d'être démasqué. En réitérant l'échange un nombre de fois suffisant, V se dira convaincu si P a toujours répondu correctement. En effet, si on fait k échanges du type précédent, la probabilité pour que P réponde correctement à tous les échanges s'il ne connaît pas la clé privée est $1/2^k$.

6.2 Cryptanalyse : définition et techniques

6.2.1 Définition

On appelle cryptanalyse la reconstruction d'un message chiffré en clair à l'aide de méthodes mathématiques. Ainsi, tout cryptosystème doit nécessairement être résistant aux méthodes de cryptanalyse. Lorsqu'une méthode de cryptanalyse permet de déchiffrer un message chiffré à l'aide d'un cryptosystème, on dit alors que l'algorithme de chiffrement a été « cassé ». On distingue habituellement quatre méthodes de cryptanalyse :

- *Une attaque sur texte chiffré seulement* consiste à retrouver la clé de déchiffrement à partir d'un ou plusieurs textes chiffrés ;
- *Une attaque sur texte clair connu* consiste à retrouver la clé de déchiffrement à partir d'un ou plusieurs textes chiffrés, connaissant le texte en clair correspondant ;
- *Une attaque sur texte clair choisi* consiste à retrouver la clé de déchiffrement à partir d'un ou plusieurs textes chiffrés, l'attaquant ayant la possibilité de les générer à partir de textes en clair ;

- Une attaque sur texte chiffré choisi consiste à retrouver la clé de déchiffrement à partir d'un ou plusieurs textes chiffrés, l'attaquant ayant la possibilité de les générer à partir de textes en clair.

6.2.2 Techniques

6.2.2.1 Méthodes cryptanalytiques

Les méthodes cryptanalytiques incluent ce qu'on appelle la cryptanalyse pratique : l'ennemi ne va pas juste contempler le cryptogramme jusqu'à ce qu'il devine le texte clair. Par exemple, il peut deviner des fragments probables du texte clair. Si le fragment est correct il peut en déduire la clé et déchiffrer le reste du message ou il peut exploiter des "isologs" le même libellé chiffré avec plusieurs systèmes ou plusieurs clés. Ainsi il peut trouver une solution même si la théorie dit qu'il n'a pas la moindre chance. Il existe plusieurs familles d'attaques cryptanalytiques, les plus connues étant l'analyse fréquentielle, la cryptanalyse différentielle et la cryptanalyse linéaire.

1- **L'analyse fréquentielle**
C'est une méthode de cryptanalyse découverte par *Al Kindi* au IXe siècle. Elle consiste à examiner la fréquence des lettres employées dans un message chiffré. Cette méthode est fréquemment utilisée pour décoder des messages chiffrés par substitution, dont un exemple très simple est le chiffre de César. Ces informations permettent aux cryptanalystes de faire des hypothèses sur le texte clair, à condition que l'algorithme de chiffrement conserve la répartition des fréquences, ce qui est le cas pour des substitutions mono-alphabétiques et poly-alphabétiques. Une deuxième condition est nécessaire pour appliquer cette technique : c'est la longueur du message à décrypter. En effet, un texte trop court ne reflète pas obligatoirement la répartition générale des fréquences des lettres. De plus, si la clé est de la même longueur que le message, il ne pourra y avoir des répétitions de lettres et l'analyse fréquentielle sera impossible.

2- **Cryptanalyse linéaire**
C'est une technique inventée par Mitsuru Matsui, chercheur chez Mitsubishi. Elle date de 1993 et fut développée à l'origine pour casser l'algorithme de chiffrement symétrique DES. Ce type de cryptanalyse se base sur un concept antérieur à la découverte de Matsui : les expressions linéaires probabilistes. La cryptanalyse linéaire consiste à faire une approximation linéaire de l'algorithme de chiffrement en le simplifiant. En augmentant le nombre de couples disponibles, on améliore la précision de l'approximation et on peut en extraire la clé. Tous les nouveaux algorithmes de chiffrement doivent veiller à être résistants à ce type d'attaque. DES n'était pas conçu pour empêcher ce genre de méthode, les tables de substitution (S-Boxes) présentent en effet certaines propriétés linéaires alors qu'elles étaient justement prévues pour ajouter une non-linéarité à DES.
Exemple : Dans une table de substitution avec 8 éléments. Soit S la fonction substitution. En

effectuant $S(X)$, on effectue une première substitution pour obtenir Y. Lors du déchiffrement, on appliquera l'opération inverse, c'est à dire $S^{-1}(Y) = S^{-1}(S(X)) = X$.

X	000	001	010	011	100	101	110	111
Y	010	100	000	111	001	110	101	011

Une telle table est non-linéaire mais la combinaison de plusieurs substitutions et opérations peut annuler en partie la non-linéarité ; c'est la faille recherchée par la cryptanalyse linéaire. Le terme linéaire se réfère en fait à une expression de la forme suivante (avec $\oplus$ l'opération binaire XOR) :

$X_1 \oplus X_2 \oplus X_3 \oplus \ldots \oplus X_N = Y_1 \oplus Y_2 \oplus Y_3 \oplus \ldots \oplus Y_N$

Le vecteur X est l'entrée et Y la sortie de la fonction que l'on essaie d'approcher avec cette équation booléenne. La variable X_i correspond à la valeur du i^{me} bit de X.

Cette expression est équivalente à :

$X_1 \oplus X_2 \oplus X_3 \oplus \ldots \oplus X_N \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus \ldots \oplus Y_N = 0$

La cryptanalyse linéaire vise à attribuer des vraisemblances aux équations possibles.

3- **Cryptanalyse différentielle**

La découverte de la cryptanalyse différentielle est généralement attribuée à Eli Biham et Adi Shamir à la fin des années 1980. Ces derniers publièrent alors un grand nombre d'attaques contre divers algorithmes de chiffrement itératif par blocs et diverses fonctions de hachage ; ces articles comprennent la présentation d'une faiblesse théorique dans l'algorithme DES. La cryptanalyse différentielle s'effectue en général dans un contexte de texte clair choisi, ce qui signifie que l'attaquant est en mesure d'obtenir les résultats chiffrés de textes clairs de son choix. Il existe des variantes qui fonctionnent dans d'autres modes d'attaque : à texte clair connu ou à texte chiffré seulement. La cryptanalyse repose sur des paires de textes clairs qui ont une différence constante. L'opération de différence peut être définie de diverses manières, la fonction OU exclusif est la plus courante. L'attaquant calcule les différences dans les textes chiffrés, afin d'en extraire des motifs pouvant indiquer un biais. Les différences en sortie du chiffrement sont nommées des différentielles. Leurs propriétés statistiques dépendent de la nature des boîtes-S et de l'algorithme de chiffrement.

4- **Cryptanalyse différentielle-linéaire**

Introduite par Martin Hellman et Susan K. Langford en 1994, la cryptanalyse différentielle-linéaire combine la cryptanalyse différentielle avec la cryptanalyse linéaire.

L'attaque différentielle sert à produire une approximation linéaire de l'algorithme, cette estimation a une probabilité de succès de 1 mais ne concerne qu'une partie du chiffrement (la probabilité de réussite serait bien inférieure sur l'ensemble du chiffrement).

6.2.2.2 Cryptanalyse basée sur les techniques d'optimisation

Plusieurs travaux impliquant des techniques d'optimisation se sont intéressés à la cryptanalyse. Des recherches cryptanalytiques utilisant l'algorithme génétique (GA) ont réussi à

attaquer les algorithmes classiques de cryptage. Une première tentative à été décrite dans [61] pour détruire l'algorithme basé sur la substitution simple.

Spillman a poursuit son travail et a montré que GA peut également être utilisé pour la cryptanalyse de cryptosystème à clé publique basé sur le problème de sac à dos [54]. Une autre tentative menée par Matthews a étudié l'utilisation du GA pour la cryptanalyse des algorithmes basés sur la transposition [62]. Dans ses travaux, la fonction fitness est basée sur la longueur de message et le nombre de diagrammes et tri grammes testés. Une recherche approfondie de la cryptanalyse de chiffrement classique a été conduite par Clark et Bagnall [63]. Les travaux de Clark [64] ont couvert une variété des algorithmes de chiffrement classiques qui incluent la simple substitution, la transposition, ainsi que le chiffrement polyalphabétiques. Il a proposé des nouvelles attaques en utilisent le recuit simulé et de la recherche tabou.

6.3 Conclusion

Dans ce chapitre, nous avons évoqué la cryptologie dans ses deux aspects : la cryptographie et la cryptanalyse en parlant de son historique, des différentes techniques utilisées. Les prochains chapitres s'intéresseront à notre propre contribution. On a choisi en connaissance de cause quatre problèmes importants et d'actualité. Le premier est la construction de fonctions courbes. Le deuxième est la construction des fonctions booléennes équilibrées de haut degré non linéarité. Le troisième (resp. quatrième) est le problème Perceptron (PP) (resp. "Permuted Perceptron" (PPP)) qui sont parmi les sujets les plus difficiles en cryptanalyse de nos jours.

Chapitre 7

Construction de fonctions courbes

7.1 Introduction

Les fonctions booléennes ou plus précisément leurs propriétés, interviennent notamment en cryptologie pour constituer des S-Boxes (tables de substitution)[65].
Une classe bien particulière de fonctions booléennes avec un nombre pair de variables est dite fonction courbe (bent dans la terminologie anglo-saxonne) si sa non-linéarité est maximale. Cela correspond à être à distance maximale (pour la distance de Hamming) de l'ensemble des fonctions booléennes linéaires, encore appelé code de Reed et Müller d'ordre 1. On dispose d'une borne générale sur la non-linéarité des fonctions booléennes, mais cette borne ne peut être atteinte que lorsque le nombre de variables est pair.
Dans ce chapitre, nous allons étudier les fonctions booléennes ayant le plus haut degré de non-linéarité qui sont les fonctions courbes. Les fonctions courbes ont été étudiées pour la première fois par Dillon[66] en 1974 et Rothaus [67] en 1976. Le mot "bent" a été utilisé par Rothaus pour la première fois. Plusieurs propriétés et constructions de fonctions courbes pourront être trouvées dans les références [[68], [69], [70], [71], [72], [73], [74]].
Nous commençons par présenter la définition et les propriétés de fonctions booléennes qui nous seront utiles pour la suite. Ensuite, nous allons nous focaliser sur les fonctions courbes en les définissant et donnant leurs propriétés basiques. Après, nous présenterons quelques techniques existantes pour la construction de fonctions courbes. A la dernière section, nous allons donner une modélisation déterministe de problème qui va être résolu par DCA. Un algorithme combiné de recuit simulé avec DCA pour la génération de fonctions courbes sera, aussi, détaillé.

7.2 Préliminaires

7.2.1 Définitions

Définition 7.2.1
Une fonction booléenne est une fonction de $\mathbb{B}^n$ dans $\mathbb{B}$ où $\mathbb{B}$ désigne le corps fini à 2 éléments $\{0,1\}$. En fait, les fonctions booléennes sont simplement un autre nom des fonctions logiques. Toutefois, lorsque l'on s'attache aux propriétés algébriques de ces fonctions, l'appellation fonction booléenne est la plus utilisée[75].

$$f : \mathbb{B}^n \to \mathbb{B}; (a_1, a_2, \cdots a_n) \longmapsto f(a_1, a_2, \cdots a_n)$$

On représente f par un tableau qui est dit Table de Vérité de f. Celle-ci indique la valeur de la fonction pour les différentes combinaisons. Il s'agit d'une table comportant une colonne pour chaque variable ainsi que pour la fonction et 2^n lignes représentant toutes les combinaisons possibles.
Par exemple : pour un nombre d'entrées $n = 2$, 1111 correspond au tableau de vérité ci-dessous.

x1	***x2***	***f***
0	*0*	*1*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

Définition 7.2.2
La fonction polaire d'une fonction booléenne $f(x)$, notée $\hat{f} = 1 - 2f(x)$, où $\hat{f} \in \{1, -1\}$.

Exemple 7.2.1
Soit f la fonction booléenne définie par $f = 10111010$.
La fonction polaire correspondante est $\hat{f} = -11 - 1 - 1 - 11 - 11$.

Définition 7.2.3
Une fonction booléenne linaire $L_w(x)$, paramétrée par w, est écrite sous la forme

$$L_w(x) = wx = w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n. \tag{7.1}$$

Définition 7.2.4
une fonction affine est définie par

$$A_w(x) = wx \oplus c \tag{7.2}$$

Définition 7.2.5
Le poids de Hamming correspond au nombre de bits différents de zéro. Il est utilisé dans plusieurs disciplines comme la théorie de l'information, la théorie des codes et la cryptographie. Pour une fonction booléenne f, le poids de Hamming est le nombre de 1 dans la table de vérité de f.

Exemple 7.2.2
Considérons la suite binaire suivante :
$f = 10111010$, *le poids de Hamming de* f, $hwt(f) = 5$.

Définition 7.2.6
La distance de Hamming doit son nom à Richard Hamming (1915-1998). Elle est décrite dans un article[76], fondateur pour la théorie des codes. Elle est utilisée en télécommunication pour compter le nombre de bits altérés dans la transmission d'un message d'une longueur donnée. La distance de Hamming est une distance au sens mathématique du terme. À deux suites de symboles de même longueur, elle associe l'entier désignant le cardinal de l'ensemble des symboles de la première suite qui différent de la deuxième. La distance Hamming entre deux fonctions booléennes est définie par le nombre de positions pour lesquelles les deux fonctions sont différentes.

Exemple 7.2.3
La distance de Hamming entre 1011101 *et* 1001001, $d(1011101; 1001001) = 2$.

Définition 7.2.7
La transformation de Fourier, appliquée aux fonctions booléennes, se révèle être un moyen très puissant pour explorer leurs différentes propriétés. Elle est, par exemple, fréquemment utilisée pour étudier des propriétés cryptographiques comme la non-linéarité maximale. On la retrouve également dans des aspects plus appliquées : l'existence d'algorithmes de calcul de la transformée de Fourier de type FFT (Fast Fourrier Transform) sert à décoder efficacement les codes de Reed et Muller. Elle est définie par :

$$\hat{F}(w) := \sum_{x} \hat{f}(x)\hat{L}_w(x); \tag{7.3}$$

7.3 Définition et propriétés de la fonction courbe

Définition 7.3.1
La fonction courbe est la fonction booléenne définie par :
$f : \mathbb{B}^n \to \mathbb{B}$.
La valeur absolue de la transformée de Walsh de f est constante. Elle est égale à $\left|\hat{F}(w)\right| := 2^{\frac{n}{2}}$
$\forall w \in \mathbb{B}^n$.

Les exemples le plus simples de fonctions courbes sont $F(x_1, x_2) = x_1x_2$ *et* $G(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4$. *Ce schéma continue :* $x_1x_2 + x_3x_4 + ... + x_{n-1}x_n$ *est une fonction courbe pour tout* n *paire[77].*

7.3.1 Propriétés :

Linéarité et Non-linéarité :
Une fonction affine est une fonction booléenne linéaire $f(x) = ax + b$ où l'addition peut être représentée par un OU-Exclusif et la multiplication par un ET. À noter que les opérations OU-Exclusif et ET produisent ensemble le même résultat que l'opération "modulo 2". Ce sont les fonctions les plus simples, hormis les constantes. L'existence d'une approximation affine d'une fonction permet, très souvent, dans le cas de chiffrements par blocs comme ceux à flot, très souvent de construire des attaques efficaces. La ressemblance en question se base sur le nombre de fois où deux fonctions prennent la même valeur. Les cryptographes utilisent le terme de non-linéarité pour parler de la distance d'une fonction booléenne à l'ensemble $\mathcal{A}$ des fonctions affines :

$$N_f = \min\{d_H(f, g) : g \in \mathcal{A}\}$$

La non-linéarité maximale possible dépend également des autres critères cryptographiques pris en compte.[78]

Propriété 7.3.1
Soit f *une fonction booléenne à* n *variables. Sa non-linéarité est égale à :*

$$N_f := 2^{n-1} - \frac{1}{2} \max_{w \in \mathbb{B}^n} |\widehat{F}(w)|. \tag{7.4}$$

Equilibre :

Définition 7.3.2
Une fonction booléenne f *à* n *variables est dite équilibrée lorsqu'elle prend autant de fois la valeur* 0 *que la valeur* 1. *Autrement dit,* f *est équilibrée si et seulement si* $hwt(f) = 2^{n-1}$

Exemple 7.3.1
La fonction $f = 0011$ *est équilibrée.*

Cette propriété est particulièrement importante en cryptographie [79][80].

Propriété 7.3.2
Rothaus a prouvé que les fonctions courbes existent uniquement pour les valeurs paires de n.

Le théorème suivant énonce que les fonctions courbes ne sont pas équilibrées.

Théorème 7.3.1
Soit f *une fonction courbe. Alors* $hwt(f) = 2^{n-1} \pm 2^{\frac{n}{2}-1}$.

Le théorème suivant résume les propriétés d'une fonction courbe :

Théorème 7.3.2
f est une fonction booléenne à n variables, les assertions suivantes sont équivalentes :

- *f est courbe,*
- *f est définie uniquement pour n pair (prouvé par Rothaus),*
- *$|\hat{F}(a)| = 2^{n/2} \forall a \in \mathbb{B}^n$,*
- *f possède un poids de Hamming égal à $2^{n-1} \pm 2^{n/2-1}$,*
- *La non-linéarité de f est égale à $2^{n-1} - 2^{n/2-1}$,*
- *f n'est pas équilibrée.*

Le théorème suivant est très important car il nous permet d'obtenir plusieurs fonctions booléennes courbes à partir d'une seule fonction courbe trouvée.

Théorème 7.3.3
Une fonction courbe reste invariante :

- *Sous une transformation linéaire ou affine de ses coordonnées, ce qui fait que f est courbe si et seulement si la fonction $h = f \circ \theta$ est courbe avec $\theta(x) = xA + \alpha$, A est une matrice d'ordre n et α un vecteur à n éléments.*
- *En ajoutant une fonction affine, c'est à dire f est courbe si et seulement si $f + \phi$ est courbe pour chaque fonction affine ϕ.*

7.3.2 Applications

Dès 1982, on découvrit que les séquences de longueur maximum basée sur les fonctions courbes ont des propriétés d'auto corrélation et de corrélation croisée aussi bonnes que celles des codes de Gold et codes Kasami pour une utilisation dans des séquences CDMA (Code Division Multiple Access)[81]. Ces séquences ont des applications larges dans le domaine de télécommunication. Les propriétés des fonctions courbes sont naturellement intéressantes en cryptographie numérique moderne. En effet, les propriétés de la fonction courbe lui donnent une certaine résistance à la cryptanalyse différentielle et à la cryptanalyse linéaire. Ces fonctions semblent être, alors, le meilleur choix pour la construction de S-Boxs ([77], [82]).

7.4 Méthodes Existantes pour la génération de fonctions courbes

7.4.1 La construction de Rothaus

En 1975, Rothaus a présenté les deux premières classes de fonctions courbes. Il a procédé par une recherche exhaustive sur toutes les fonctions booléennes à 6 variables [67]. En

considérant la forme algébrique de la fonction, Rothaus s'est rendu compte que la recherche exhaustive est possible car on peut ramener les fonctions à 6 variables aux formes suivantes en appliquant une transformation linéaire :

1. $x_1x_2x_3$
2. $x_1x_2x_3 + x_4x_5x_6$
3. $x_1x_2x_3 + x_2x_4x_5$
4. $x_1x_2x_3 + x_2x_4x_5 + x_3x_4x_6$.

Deux classes ont été présentées par Rothaus dans [67]

Théorème 7.4.1
Soit $n = 2k$ et x, y deux vecteurs ayant k éléments et f une fonction arbitraire dans $\mathbb{B}^k$. La fonction $Q(x, y) \in \mathbb{B}^{2k}$ définie par $Q(x, y) = x_1y_1 + x_2y_2 + \cdots + x_ky_k + f$ est courbe.

Théorème 7.4.2
Soient A(x), B(x), C(x) des fonctions courbes dans $\mathbb{B}^{2k}$ telle que $A(x)+B(x)+C(x)$ est aussi courbe. Soient $y, z \in V_1$, (V_1 l'espace vectoriel composé des séquences de taille 1 dont les composants sont dans $GF(2)$). La fonction $Q(x, y, z) = A(x)B(x) + B(x)C(x) + C(x)A(x) + [A(x) + B(x)]y + [A(x) + C(x)]z + yz$.

7.4.2 La construction de Carlet

Carlet a utilisé des fonctions courbes connues et il les a modifiées pour obtenir des nouvelles fonctions [68]. Il a introduit deux classes de fonctions courbes en modifiant la classe de Maiorana McFarland. Le théorème suivant décrit la méthode de construction.

Théorème 7.4.3
Soit E un sous espace linéaire de $(\frac{n}{2})$de $\mathbb{B}^n$ et π une permutation sur $\mathbb{B}^{\frac{n}{2}}$ telle que, pour tout (x, y) in E, le nombre : $x.\pi(y)$ est égal à 0. Alors le fonction définie sur $\mathbb{B}^n$ comme étant

$$f(x, y) = x.\pi(y) + \phi_E(x, y).$$

est courbe, avec ϕ_E est la fonction de support E.

7.4.3 La construction de Maiorana McFarland

La classe de fonction courbe trouvée par McFarland est une extension de celle trouvée par Rothaus.

Théorème 7.4.4
Soit k un entier positif choisi au hasard et $n = 2k$. La fonction $f \in \mathbb{B}^n$ est définie par $f(x) = x_2 \cdot \pi(x_1) + g(x_1)$ avec x_1, x_2 deux vecteurs à k éléments, x est alors définie comme suit $x = [x_1, x_2]$, π est une permutation arbitraire et $g \in \mathbb{B}^k$ est une fonction arbitraire. La fonction f est courbe.

Nous avons utilisé des fonctions courbes obtenus par la méthode de Mariona McFarland comme point initial pour l'algorithme de génération de fonctions booléennes équilibrées de haut degré de non-linéarité [83].

7.4.4 Génération aléatoire de fonctions courbes

L'algorithme pour la génération de fonctions courbes dans le domaine ANF prend comme entrée le nombre maximum et minimum de coefficients ANF que la fonction résultante peuvent avoir. Puisque l'ordre de non-linéarité de fonctions courbes est inférieur ou égal à $n/2$, visiblement aucun coefficient ANF ne pourra être supérieur à $n/2$. Cette restriction est la raison principale pour la faisabilité de la génération aléatoire, car elle réduit considérablement l'espace de recherche possible [84].

7.5 Modélisation DC et l'algorithme SA-DCA pour la génération de fonctions courbes

7.5.1 Formulation mathématique

On considère le théorème suivant :

Théorème 7.5.1
Soit f une fonction booléenne. f est considérée comme courbe si et seulement si $d(f, A_n) = N_{max}$ avec $N_{max} = 2^{n-1} - 2^{\frac{n}{2}-1}$

Nous avons,

$$N_{max} = 2^{n-1} - \tfrac{1}{2} \min_{f \in F} \phi(\hat{f}) = 2^{n-1} - 2^{\frac{n}{2}-1},$$

avec

$$\phi(\hat{f}) := \max_{w \in \mathbb{B}^n} | \sum_{x \in \mathbb{B}^n} \widehat{f}(x) \widehat{L}_w(x)|$$

Pour la définition de $\widehat{L}_w$, nous pouvons remarquer la présence de l'opérateur XOR qui ne peut pas être utilisé dans un problème d'optimisation continu. C'est pour cela qu'on a montré par récursivité que

$$\begin{aligned} \widehat{L}_w(x) &= 1 - 2wx = 1 - 2(w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n.) \\ &= \begin{cases} 1 \ si \langle w, x \rangle = \sum_{i=1}^{n} w_i x_i \ est \ un \ nombre \ pair, \\ -1 \ sinon \end{cases} \end{aligned}$$

Pour tout $w, x \in \mathbb{B}^n$, notons : $a_{w,x} := \widehat{L}_w(x) \in \{-1, 1\}$.
N_{max} s'écrit aussi

$$N_{max} = 2^{n-1} - \min_{x \in \mathbb{B}^n} \Psi(f)$$

avec

$$\Psi(f) := \max_{w \in \mathbb{B}^n} \left| \sum_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in \mathbb{B}^n} a_{wx} \right| = 2^{\frac{n}{2}-1}$$

Nous désignons par K le domaine de solutions réalisables de notre problème :

$$K = \left\{ \Psi(f) \leq 2^{\frac{n}{2}-1}, 0 \leq f_x \leq 1, \ \forall x \in \mathbb{B}^n \right\}$$

Notre problème est équivalent au problème suivant

$$0 = \min \left\{ p(f) := \sum_{x \in \mathbb{B}^n} min\{f_x, 1 - f_x\}, f_x \in K \right\} \tag{7.5}$$

Il est clair que p est une fonction concave non négative sur $[0, 1]^n$. En plus $p(f) = 0$ si $f \in \{0, 1\}^n$. Ainsi notre problème peut être exprimé de la façon suivante :

$$(Q) \quad \beta = min \left\{ p(f), \Psi(f) \leq 2^{\frac{n}{2}-1}, 0 \leq f_x \leq 1, \ \forall x \in \mathbb{B}^n \right\}$$

Si la fonction obtenue est courbe, $\beta = 0$ et $\Psi(f) = 2^{\frac{n}{2}-1}$, le problème considéré est, alors, une minimisation d'une fonction quadratique concave à contraintes linéaires. Pour ces types de problèmes, DCA a été intensivement développé.

7.5.2 La formulation DC

Soit χ_Q la fonction indicatrice de Q définie comme suit :

$$\begin{cases} 0 \ si \ f_x \in K, \\ +\infty \ sinon. \end{cases} \tag{7.6}$$

Une décomposition naturelle de la fonction objective serait

$$Obj_f := G(f) - H(f),$$

avec

$$G(f) = \chi_Q et H(f) = -p(f) := -(\sum_{x \in \mathbb{B}^n} min\{f_x, 1 - f_x\}). \tag{7.7}$$

Les fonctions G et H sont convexes donc le programme est DC. Selon la description de DCA, la résolution de (Q) par DCA consiste à déterminer deux suites x^l et y^l telles que :

$$y^l \in \partial H(x^l),$$
$$x^{l+1} \in \partial G(y^l).$$

Par définition de H, son sous gradient peut s'écrire comme

$$y_i{}^l = \begin{cases} -1 \ six_i{}^l < 0, \\ 1 \ sinon; \end{cases} \tag{7.8}$$

Calculer $x^{l+1} \in \partial G(y^l)$, revient à calculer

$$x^{l+1} = argmin \left\{ G(x) - \langle y^l, x \rangle : x \in K \right\}. \tag{7.9}$$

x^{l+1} est, alors, solution optimale de problème linéaire

$$min \left\{ -\langle y^l, x \rangle : x \in K \right\}. \tag{7.10}$$

7.5.3 Schéma DCA

Nous pouvons maintenant décrire le schéma de DCA pour la résolution du problème (7.5).

Algorithme 7.5.1
Soit $f^0 \in \mathbb{R}^{2^n}$, et soit ϵ un nombre suffisement petit.
Répéter
Soit $v^k \in \partial H(f)$ par la formule (7.8) ;
Résoudre le programme linéaire (7.10) pour obtenir f^{k+1} ;
$k := k + 1$.
Jusqu'à $\| f^k - f^{k-1} \| < \epsilon$.

7.5.4 L'approche hybride SA-DCA

Vu le nombre de minimas locaux du problème, l'utilisation de DCA tout seul pour la résolution pousse rapidement l'algorithme à se trouver bloqué dans un minimum local. Nous avons eu l'idée, donc, de combiner DCA qui a toujours prouvé qu'il pourra atteindre l'optimum si nous commençons par un "bon" point de départ tout en garantissant un temps de calcul minimum avec le recuit simulé. Ce dernier a toujours été la bonne solution pour échapper aux minimas locaux en acceptant même les mauvaises solutions dans un objectif d'exploration maximale de l'espace de recherche.
Afin de tirer profit des avantages des algorithmes : recuit simulé et DCA, nous avons implémenté un algorithme combiné qui suit le schéma SA-DCA décrit dans (5.2.2).

7.5.4.1 choix de la solution initiale

Comme avec toutes les heuristiques, particulièrement à des fins d'éfficacité, l'étape la plus importante est de toujours commencer par une "bonne" solution. Elle devra être une fonction ayant une valeur importante de non-linéarité pour qu'elle puisse arriver à travers les différentes étapes à conduire notre algorithme vers une fonction courbe. Dans notre approche, nous avons choisi deux cas de figures :

- Commencer par une fonction courbe à n variables pour obtenir une nouvelle fonction courbe ayant le même nombre de variables.
- Commencer par une fonction aléatoire à n variables mais ayant la valeur maximale de non-linéarité pour obtenir une fonction courbe à n variables.

7.5.4.2 Description de l'algorithme SA-DCA

L'algorithme comporte deux procédures : recuit simulé et DCA. C'est le recuit simulé qui prend la main pour construire des fonctions booléennes ayant un haut degré de non-linéarité, obtenu en minimisant la valeur de $\Psi(f)$. Si nous continuons à tourner le recuit simulé, à la fin, nous finirons par obtenir une fonction avec une valeur importante de non-linéarité sans aucune garantie que la fonction soit courbe. Cette contrainte est désormais contrôlée par la procédure DCA qui va utiliser Cplex pour résoudre le problème (7.5). L'algorithme est implémenté suivant le schéma SA-DCA décrit dans le chapitre "Approches hybrides".

7.6 Résultats numériques

Tout au long de la deuxième partie de cette thèse, nos algorithmes sont implémentés en C++ et testés sur une machine équipée d'un processeur *AMD Athlon 64 bits,dual core 3800+ (processeur). Le logiciel Cplex 9.5* a été utilisé pour résoudre les programmes linéaires.
Les valeurs des paramètres utilisés dans le recuit simulé (SA) doivent être choisies avec soin puisqu'elles peuvent avoir une influence significative sur la performance de l'algorithme. Toutefois, il n'est pas facile de trouver les bonnes valeurs des paramètres parce qu'un large éventail de valeurs devra être considéré pour chaque paramètre et certains paramètres peuvent être mis en corrélation les uns avec les autres. Dans nos travaux, ces paramètres sont choisis en utilisant des modèles expérimentaux.
Nous avons d'abord généré au hasard à partir de notre espace de recherche un nombre de fonctions que nous allons utiliser pour déterminer l'éventail des valeurs de ΔE qui seront rencontrées lors de notre déplacement. Le paramètre T est choisi de telle façon qu'il soit considérablement plus grand que la plus grande valeur de ΔE normalement rencontrée. Nous procédons, après, à la diminution de la température T en utilisant un facteur $\alpha = 0.9$. La valeur de T est maintenue constante pendant 100 configurations. Si nous n'arrivons pas à trouver la fonction courbe espérée, nous abandonnons au-delà de 40 configurations. La

fonction objectif utilisée est :

$$\psi(f) = \max_{w \in \mathbb{B}^n} \left| \sum_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in \mathbb{B}^n} a_{wx} \right|.$$

Elle est définie dans le chapitre "Génération de fonctions booléennes de haut degré de non linéarité", à la section "Approche déterministe (DCA)".
Pour l'implémentation de DCA, $\epsilon = 10^{-6}$.
Le tableau (7.1) donne des exemples de fonctions courbes (f) obtenues associées au temps d'exécution nécessaire pour les obtenir par SA-DCA (nous considérons les valeurs de la variable n 8,10 et 12). Quelques caractéristiques cryptographiques (immunité à la corrélation (IC) et degré algébrique (d)) sont aussi considérées.

7.7 Conclusion

Dans ce chapitre, nous avons étudié les fonctions courbes du point de vue cryptographique.
Après avoir présenté les propriétés les plus importantes de fonctions courbes, nous avons donné quelques techniques de génération existantes. Pour finir, nous avons décrit notre approche pour la génération de fonctions courbes : le problème fonctions courbes à été modélisé comme un modèle déterministe et résolu par DCA. Nos contributions propres portent à la fois sur la modélisation et les méthodes de résolution.

n	f	d	IC	temps CPU
8	00FF000069965AA533CC5A5A33330FF055AA3C3C666 63CC3555569690F0F6699	4	0	5min 19sec
10	6A6A6A7B6A956A843F3F3F2E3FC03FD16A6A958 46A95957B3F3FC0D13FC0C02E6A6A6A7B6A956A8 43F3F3F2E3FC03FD16A6A95846A95957B3F3FC0D1 3FC0C02E6A6A6A7B6A956A843F3F3F2E3FC03FD16 A6A95846A95957B 3F3FC0D13FC0C02E95959584956 A957BC0C0C0D1C03FC02E95956A7B956A6A84C0C03 F2EC03F3FD1	4	0	17min 19sec
12	00CC00CCAA66AA66FF33FF3355995599FF33FF33559 95599FF33FF335599559900CC00CCAA66AA66FF33FF 3355995599FF33FF3355995599FF33FF3355995599F03C F03C5A965A960FC30FC3A569A5690FC30FC3A569A5 690FC30FC3A569A569F03CF03C5A965A960FC30FC3A 569A5690FC30FC3A569A5690FC30FC3A569A569FF33 FF335599559900CC00CCAA66AA6600CC00CCA A66AA6600CC00CCAA66AA6600CC00CCAA66A A66FF33FF3355995599FF33FF3355995599FF33FF33 559955990FC30FC3A569A569F03CF03C5A965A96F0 3CF03C5A965A96F03CF03C5A965A96F03CF03C5A 965A960FC30FC3A569A5690FC30FC3A569A5690FC 30FC3A569A56900CCFF33AA665599FF3300CC5599AA 66FF3300CC5599AA66FF3300CC5599AA6600CCFF33 AA665599FF3300CC5599AA66FF3300CC5599AA66FF 3300CC5599AA66F03C0FC35A96A5690FC3F03CA5695 A960FC3F03CA5695A960FC3F03CA5695A96F03C0FC 35A96A5690FC3F03CA5695A960FC3F03CA5965A690 FC3F03CA5965A69FF3300CC5599AA6600CCFF33AA 66559900CCFF33AA66559900CCFF33AA66559900CC FF33AA665599FF3300CC5599AA66FF3300CC5599AA 66FF3300CC5599AA660FC3F03CA5695A96F03C0FC3 5A96A569F03C0FC35A96A569F03C0FC35A96A569F03 C0FC35A96A5690FC3F03CA5695A960FC3F03CA5965 A690FC3F03CA5965A69	6	0	0h 57min 43sec

TABLE 7.1 – Les fonctions courbes obtenues pour n=8, 10 et 12

Chapitre 8

Génération de fonctions booléennes de haut degré de non linéarité

8.1 Introduction

L'étude des fonctions booléennes est liée à de nombreux sujets comme la théorie algébrique des codes, la théorie des séquences, la théorie des designs et, plus récemment, la cryptographie. En fait, les fonctions booléennes sont devenues importantes pour la conception et la sécurité de certains systèmes de chiffrement symétrique : les systèmes de chiffrement par blocs et de chiffrement à flot. Les fonctions utilisées dans ces systèmes doivent avoir des propriétés très particulières pour résister à des attaques spécifiques dont les principales sont la cryptanalyse linéaire, la cryptanalyse différentielle, l'attaque par corrélations et les attaques algébriques. Les attaques mises au point à la fin des années 1980 et au début des années 1990 ont, en effet, pu être formalisées en mettant en évidence le rôle des fonctions booléennes en cryptographie et les propriétés qu'elles doivent vérifier, dont : haute non-linéarité, équilibre, immunité aux corrélations, degré élevé et bien d'autres. Ces propriétés ont été largement étudiées dans la littérature ([85], [86] et les références incluses).
Le but de ce chapitre est de présenter une nouvelle approche hybride pour la génération de fonctions booléennes de haut degré de non-linéarité. En effet, l'approche consiste à combiner une approche déterministe basée sur la programmation DC et une approche probabiliste basée sur l'algorithme de recuit simulé.
Nous commençons par décrire brièvement les critères cryptographiques des fonctions booléennes auxquelles nous allons nous intéresser.
Ensuite, nous allons présenter certaines approches existantes dans la littérature. Pour finir, notre approche est détaillée et les résultats numériques sont commentés.

8.2 Les propriétés cryptographiques d'une fonction booléenne

Dans ce chapitre, nous nous intéressons au rôle joué par les fonctions booléennes en cryptographie. Compte tenu des principes de confusion et diffusion et des nombreuses attaques publiées sur des constructions existantes, des critères de résistance ont été définis afin d'être satisfaits au mieux lors de l'élaboration d'un système de chiffrement et du choix des fonctions booléennes cryptographiques à utiliser. Cependant tous les critères ne vont pas dans le même sens et de nombreux compromis peuvent être nécessaires. Nous nous intéressons dans ce chapitre aux deux propriétés : non-linéarité et équilibre.

8.3 Méthodes existantes pour la génération de fonctions booléennes de haut degré de non linéarité

La quête de bonnes fonctions booléennes cryptographiques a mené à différentes approches. Certaines de ces approches sont basées sur les méthodes méta heuristiques de l'optimisation, d'autres sur les méthodes déterministes et d'autres ont fait l'hybridation entre différentes approches ([87], [88], [89], [90] et les références incluses). Dans cette section, nous allons présenter quelques techniques trouvées dans la littérature.

8.3.1 Méthode de Hill Climbing

Il s'agit de modifier légèrement les entrées d'une fonction booléenne donnée pour améliorer certaines caractéristiques, par exemple la non-linéarité ([91][64]). La méthode de Hill Climbing consiste à améliorer les caractéristiques d'une fonction booléenne en appliquant des altérations sur les entrées de la table de vérité à des positions bien déterminées. Le nombre de ces altérations peut être un ou deux. Chaque modification entraîne celle de la transformée de Walsh. En effet, si on note Δ_{WHT} cette variation, on va trouver que $\Delta_{WHT} \in \{2, -2\}$ si on fait l'altération en une seule position et $\Delta_{WHT} \in \{4, 0, -4\}$ si on la fait à deux positions. Etant donnée une fonction booléenne $f(x)$ et WHT $= \left|\hat{F}(w)\right|$, l'algorithme consiste à commencer par définir les ensembles suivants :

- $W_{1^+} = \left\{w, \hat{F}(w) = WH_{max}\right\}$,
- $W_{1^-} = \left\{w, \hat{F}(w) = -WH_{max}\right\}$,
- $W_{2^+} = \left\{w, \hat{F}(w) = WH_{max} - 2\right\}$,
- $W_{2^-} = \left\{w, \hat{F}(w) = -(WH_{max} - 2)\right\}$,

- $W_{3^+} = \left\{w, \hat{F}(w) = WH_{max} - 4\right\}$,
- $W_{3^-} = \left\{w, \hat{F}(w) = -(WH_{max} - 4)\right\}$,
- $W^+ = W_1^+ \cup W_2^+$,
- $W^- = W_1^- \cup W_2^-$.

Une entrée x peut être un élément de la liste des améliorations si les conditions suivantes sont vérifiées :

- $f(x) = L_w(x) \, \forall w \in W^+$,
- $f(x) \neq L_w(x) \, \forall w \in W^-$.

Améliorer la non-linéarité d'une fonction équilibrée est l'un des objectifs qu'on veut atteindre lors de l'application de Hill Climbing.
Pour arriver à réaliser cela, on va effectuer un nombre pair de changements sur la table de vérité. En effet, en complémentant deux entrées x_a et x_b, on peut augmenter la non-linéarité tout en conservant le poids de Hamming. On effectue les modifications selon le schéma suivant :
On définit :

- $W_1 = W_{1^+} \cup W_{1^-}$,
- $W_{2,3^+} = W_{2^+} \cup W_{3^+}$,
- $W_{2,3^-} = W_{2^-} \cup W_{3^-}$.

Une paire d'entrées (x_a, x_b) fait partie de la liste des améliorations si les conditions suivantes sont vérifiées :

- $f(x_a) \neq f(x_b)$,
- $L_w(x_a) \neq L_w(x_a) \, \forall w \in W_{1^+}$,
- $f(x_i) = L_w(x_i), i \in \{1,2\} \, \forall w \in W_{1^-}$,
- $f(x_i) \neq L_w(x_i), i \in \{1,2\} \, \forall w \in W_{1^-}$,
- $\forall w \in W_{2;3}^+, si L_w(x_a) \neq L_w(x_b) \;\; donc \; f(x_i) = L_w(x_i), i \in \{1,2\}$,
- $\forall w \in W_{2;3}^-, si L_w(x_a) \neq L_w(x_b) \;\; donc \; f(x_i) \neq L_w(x_i), i \in \{1,2\}$.

L'algorithme risque de se trouver piégé dans un optimum local ou bien de se bloquer s'il ne trouve pas des résultats meilleurs que le résultat actuel (c'est à dire s'il ne trouve pas

les entrées qui vérifient les règles ci-dessus). Selon les résultats de [91][64], la méthode de Hill Climbing permet de bien améliorer la non-linéarité de non seulement les fonctions booléennes générées aléatoirement mais aussi les fonctions booléennes générées par l'algorithme génétique.

8.3.2 Approche Hybride : Hill Climbing et algorithme génétique

Cette approche consiste à combiner une approche heuristique (algorithme génétique) et la méthode de Hill Climbing. En effet, l'approche heuristique, telle que, décrite dans [64], a été utilisée pour améliorer les résultats de l'algorithme de Hill Climbing. En partant d'une génération choisie aléatoirement, on va essayer d'améliorer la non-linéarité d'une génération à l'autre à travers une fonction de sélection, de reproduction et une autre de mutation pour imiter la reproduction sexuée des êtres vivants. L'application ou non de l'algorithme de Hill Climbing est décidée par une variable booléenne dans l'algorithme. La nouvelle descendance est alors ajoutée et on passe à l'étape suivante de l'algorithme génétique. Cette approche a fourni de bons résultats comparés à ceux de la méthode de Hill Climbing seule et ceux de la génération aléatoire des fonctions booléennes[64].

Une autre approche hybride utilisée consiste à combiner le recuit simulé avec le hill Climbing. Le recuit simulé est une approche heuristique qui permet d'éviter de se trouver piégé dans un optimum local en offrant une exploration plus large de l'espace de recherche. Vu qu'une fonction d'évaluation est indispensable dans un problème d'optimisation, on a choisi dans la formulation de cette approche une fonction coût qui vise à améliorer la non-linéarité. Après avoir obtenu la plus petite valeur de fonction coût selon un paramétrage bien déterminé, on applique l'algorithme de Hill Climbing pour aboutir à un optimum local qui vérifie une bonne valeur de non-linéarité. A chaque étape, on détermine le couple (non-linéarité, auto correction). Les résultats expérimentaux ont montré que pour des petites valeurs de variables des fonctions booléennes, l'algorithme donne toujours de bons résultats. Par contre pour des grandes valeurs, l'algorithme de recuit simulé devient très lent ([90] [86]).

8.3.3 Approche déterministe (DCA)

C'est une approche développée au sein de LITA ([92][1]) qui consiste à formuler le problème comme un problème DC polyédral à variables mixtes 0-1. Les relations suivantes expriment les contraintes ainsi que le problème qui doit être résolu afin d'obtenir des fonctions booléennes cryptographiques.

i. Le domaine des solutions réalisables est

$$\begin{cases} |\sum_x f_x - 2^{n-1}| \leq b \text{ avec } b \geq 0, \\ 0 \leq f_x \leq 1, \forall x \in B^n. \end{cases}$$

ii. La non linéarité d'une fonction booléenne f est exprimée par

$$\max N_f = 2^{n-1} - \min_{w \in \mathbb{B}^n} \psi(f),$$

avec $\psi(f) = \max_{w \in \mathbb{B}^n} \left| \sum_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in \mathbb{B}^n} a_{wx} \right|$ qui est une fonction polyédrale convexe. Comme le problème est à variables discrètes, la technique de pénalité exacte a été utilisée. Le problème $\mathcal{Q}$ à résoudre est alors

$$\begin{cases} \beta = \min \psi(f) + \text{tp(f)}, \\ s.t \\ |\sum_x f_x - 2^{n-1}| \leq b \text{ avec } b \geq 0, \\ 0 \leq f_x \leq 1, \forall x \in \mathbb{B}^n. \end{cases}$$

La résolution de ce problème a été effectuée en utilisant DCA (Algorithme de différence de deux fonctions convexes).
Les résultats fournis sont très encourageants en les comparant avec ceux d'un algorithme hybrides (Génétique et Hill Climbing) et ceux d'une génération aléatoire [92].

8.4 Formulation mathématique du problème de génération de fonctions booléennes équilibrées de haut degré de non linéarité [1]

Comme on l'a déjà évoqué à la section précédente, le choix d'une bonne fonction booléenne est un problème difficile et en pratique le concepteur d'un système de chiffrement a un choix relativement restreint. Il convient en effet d'utiliser des fonctions avec de bonnes propriétés pour résister aux nombreuses attaques découvertes au cours de l'évolution de la cryptographie. L'objectif de cette section est de présenter brièvement la modélisation mathématique du problème.
D'après (7.3 et 7.4), nous avons,

$$N_f = \min_{w \in \mathbb{B}^n} \frac{1}{2} \left(2^n - | \sum_{x \in \mathbb{B}^n} \widehat{f}(x) \widehat{L}_w(x) | \right), \tag{8.1}$$

En posant

$$\phi(\hat{f}) := \max_{w \in \mathbb{B}^n} | \sum_{x \in \mathbb{B}^n} \widehat{f}(x) \widehat{L}_w(x) |, \tag{8.2}$$

(8.1) devient

$$\max_{f\in F} N_f = 2^{n-1} - \frac{1}{2}\min_{f\in F}\phi(\hat{f}). \tag{8.3}$$

avec ϕ s'écrit ainsi

$$\phi(\hat{f}) := \max_{w\in\mathbb{B}^n} |\sum_{x\in\mathbb{B}^n} \widehat{f}(x)a_{wx}|. \tag{8.4}$$

en remplaçant $\widehat{f}$ par sa valeur $(2f_x - 1)$, nous obtenons

$$\phi(\hat{f}) := \max_{w\in\mathbb{B}^n} |\sum_{x\in\mathbb{B}^n} a_{wx}(2f_x - 1)| = 2\max_{w\in\mathbb{B}^n} |\sum_{x\in\mathbb{B}^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in\mathbb{B}^n} a_{wx}|. \tag{8.5}$$

Ce qui donne pour (8.1)

$$\max_{f\in F} N_f = 2^{n-1} - \min_{x\in\mathbb{B}^n}\Psi(f), \tag{8.6}$$

avec

$$\Psi(f) := \max_{w\in\mathbb{B}^n}\left|\sum_{x\in\mathbb{B}^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in\mathbb{B}^n} a_{wx}\right|. \tag{8.7}$$

Ainsi, maximiser N_f revient à minimiser $\Psi(f)$ dans $\mathbb{B}^n$.
Pour trouver des fonctions booléennes équilibrées, la contrainte suivante s'impose

$$|\sum_{x\in\mathbb{B}^n} f_x - 2^{n-1}| \leq b, \tag{8.8}$$

avec un nombre positif b. Visiblement si $b = 0$, la fonction est équilibrée.
Le problème a été finalement modélisé dans [[1], [93]] de la façon suivante :

$$\beta := \min\{\Psi(f) : 2^{n-1} - b \leq \sum_{x\in\mathbb{B}^n} f_x \leq 2^{n-1} + b,\ f \in \mathbb{B}^n\}. \tag{8.9}$$

8.5 Résolution de problème par la programmation DC et DCA

Afin de résoudre notre problème $(\mathcal{Q})$ par la programmation DC, une première étape consiste à l'écrire comme étant un programme DC. Nous considérons K, l'ensemble de solutions réalisables de $(\mathcal{Q})$.

$$K := \left\{ f : 2^{n-1} - b \leq \sum_{x\in\mathbb{B}^n} f_x \leq 2^{n-1} + b,\ 0 \leq f_x \leq 1,\ \forall x \in \mathbb{B}^n \right\}.$$

Comme les variables de notre problème sont discrètes, nous avons utilisé la technique de pénalité exacte pour rendre le problème à variables continus.
Le théorème de pénalité exacte suivant est considéré :

Théorème 8.5.1
Soit S un ensemble convexe polyédral borné et non vide, f une fonction concave finie sur S et p une fonction concave finie et non négative sur S. Alors il existe $t > 0$ tel que les deux problèmes suivants $(\mathcal{P})$ et $(\mathcal{P}_t)$ sont équivalents.

$$(\mathcal{P}_t)\ \alpha(t) = \inf\{f(x) + tp(x) : x \in S\}$$

$$(\mathcal{P})\ \alpha = \inf\{f(x) : x \in S, p(x) \leq 0\}$$

Plus précisément si $V(S) = \{x \in S, p(x) \leq 0\}$ *alors* $t_0 = 0$,
sinon $t_0 = \min\left\{\frac{f(x)-\alpha(0)}{\beta}, x \in S : p(x) \leq 0\right\}$
où $\beta = \min\{p(x) : x \in V(S), p(x) > 0\} > 0$.

La fonction de pénalité p est définie par

$$p : \mathbb{R}^{2^n} \to \mathbb{R} \text{ est la fonction définie par } p(f) := \sum_{x \in \mathbb{B}^n} \min\{f_x, 1 - f_x\}$$

Il est clair que :

- la fonction est concave et positive sur $[0,1]^n$
- et que $p(f) = 0$ *ssi* $f \in \mathbb{B}^n$

Soit χ_K la fonction indicatrice de K avec $\chi_K(f) = 0$ si $f \in K$ sinon $+\infty$. Comme K est convexe, χ_K est une fonction convexe dans $\mathbb{R}^{2^n}$.

Une décomposition DC de la fonction objective de $(\mathcal{Q})$ s'impose :

$$\Psi(f) + tp(f) := G(f) - H(f),$$

avec $G(f) := \chi_{K(f)} + \Psi(f)$ et $H(f) := -tp(f)$.
Il est clair que G et H sont des fonctions convexes, le problème $(\mathcal{Q})$ est alors un problème DC ayant la forme suivante :

$$(Q_{dc}) \quad \beta := \min\{G(f) - H(f) : \quad f \in \mathbb{R}^{2^n}\}.$$

Soit ψ_w la fonction définie par : $\psi_w(f) := \left|\sum_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in \mathbb{B}^n} a_{wx}\right|$. Alors ψ_w est une fonction convexe, et $\Psi(f) = \max_{w \in \mathbb{B}^n} \psi_w(f)$. Par ailleurs, Ψ est une fonction polyédrale convexe. La fonction H est aussi polyédrale convexe.

8.5.1 DCA pour la résolution de $(\mathcal{Q})$

Appliquer DCA au problème $(\mathcal{Q})$ revient à calculer à chaque itération k : $v^k \in \partial H(f^k)$ et $f^{k+1} \in \partial G^*(v^k)$.
D'après la définition de H, nous pouvons prendre v^k comme suit :

$$v_x^k := -t \quad \text{si} \quad f_x^k \leq 0.5, \; et \; t \quad \text{sinon.} \tag{8.10}$$

D'autre part, calculer $f^{k+1} \in \partial G^*(v^k)$ revient à trouver la solution du problème linéaire suivant :

$$\min \left\{ \begin{array}{c} \xi - \langle v^k, f \rangle : \sum\limits_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum\limits_{x \in B^n} a_{wx} \leq \xi, \forall w \in \mathbb{B}^n \\ - \sum\limits_{x \in \mathbb{B}^n} a_{wx} f_x + \frac{1}{2} \sum\limits_{x \in \mathbb{B}^n} a_{wx} \leq \xi, \forall w \in \mathbb{B}^n, \\ 2^{n-1} - b \leq \sum\limits_{x \in \mathbb{B}^n} f_x \leq 2^{n-1} + b, 0 \leq f_x \leq 1, \; \forall x \in \mathbb{B}^n \end{array} \right\}. \tag{8.11}$$

L'algorithme DC à appliquer sur notre problème est le suivant :

Algorithme 8.5.1
Soit $f^0 \in \mathbb{R}^{2^n}$, et soit ϵ un nombre suffisement petit.
Répéter
Soit $v^k \in \partial H(f)$ par la formule (8.10) ;
Résoudre le programme linéaire (8.11) pour obtenir f^{k+1} ;
$k := k + 1$.
Jusqu'a $\| f^k - f^{k-1} \| < \epsilon$.

Soient Ω l'ensemble de solutions réalisables du problème linéaire (8.11) et$V(\Omega)$ l'ensemble de sommets. Soit f^* une solution trouvée par **DCA**. La convergence de **DCA** se résume au théorème suivant dont la démonstration est basée sur le théorème de convergence de DCA appliqué à un programme DC polyédral :

Théorème 8.5.2
*(Les propriétés de convergence de l'algorithme **DCA**)*
*(i) **DCA** génére une séquence $\{f^k\}$ contenue dans $V(\Omega)$ tel que la séquence $\{\Psi(f^k) + tp(f^k)\}$ soit décroissante.*
(ii) Pour un nombre t suffisamment large, si à une itération r on a $f^r \in \{0,1\}^{2^n}$, alors $f^k \in \{0,1\}^{2^n}$ pour tout $k \geq r$.
(iii) La séquence $\{f^k\}$ converge vers $\{f^*\} \in V(\Omega)$ après un nombre fini d'itérations. Le point f^* est considéré comme un point critique du problème (Q_{dc}). En plus si $f_x^* \neq \frac{1}{2}$ pour tout $x \in \mathbb{B}^n$, alors f^* est une solution locale de (Q_{dc}).

8.6 Nos approches hybrides pour la génération de fonctions booléennes équilibrées de haut degré de non-linéarité

8.6.1 L'approche hybride SA-DCA

Afin de tirer profit des avantages des algorithmes recuit simulé et DCA, nous avons implémenté un algorithme combiné qui suit le schéma SA-DCA décrit dans le chapitre "Approches hybrides" [94].

Choix de point initial pour SA-DCA

Dans le chapitre précédant, la classe de fonctions booléennes ayant le haut degré de non-linéarité a été étudiée : les fonctions booléennes courbes. Plusieurs algorithmes ont été proposés pour transformer ces fonctions en des fonctions équilibrées en essayant de maintenir un haut degré de non-linéarité.
Dans cette perspective et pour les valeurs paires de n nous avons essayé de recommencer notre algorithme SA-DCA avec une fonction booléenne courbe pour essayer de la rendre équilibrée tout en maintenant un haut degré de non-linéarité.

8.6.2 L'approche hybride CE-DCA

Dans cette section, nous allons opter pour une combinaison de l'algorithme d'entropie croisée avec l'algorithme DCA. Pour mettre en oeuvre l'algorithme, nous avons besoin d'un système de représentation de solutions candidates et d'un mécanisme aléatoire pour les générer.

Pour notre algorithme CE, nous avons choisi de représenter les solutions candidates par un vecteur binaire de longueur n, de telle sorte que tous les vecteurs X disposent d'une représentation unique $X = (X_1, X_2, ..., X_n)$, où $X_i \in (0, 1)$ [95].

Afin de générer ces solutions candidates, l'algorithme CE maintient et met à jour un ensemble de paramètres de référence $\left\{p_{(t,i)} \in [0,1], i = 1, \cdots, n\right\}$, avec $(t \in \mathbb{N})$ est le compteur d'itérations. Pour chaque valeur de t, les référence $p_{(t,i)}$ ont été collectées dans un vecteur de référence, p_t. Les solutions candidates sont obtenues par génération de vecteurs aléatoires de la forme $X = (X_1, X_2, ..., X_n)$, où les $\{X_i, i = 1, \cdots, n\}$, sont des variables indépendantes de Bernoulli ayant respectivement les paramètres $p_{t,i}$.

Ainsi, le vecteur p_t est paramétrable par la fonction de probabilité $f : \{0,1\}^n \to [0,1]$, définie par

$$f(x; p_t) = \prod_{i=1}^{n} {p_{t,i}}^{x_i} (1 - p_{t,i})^{(1-x_i)}.$$

Notre algorithme prend en entrée les paramètres suivants :

- un vecteur de référence initial p_0 qui vérifie $0 < p_{0,i} < 1, i = \{1, \cdots, n\}$,
- un entier positif N, pour préciser le nombre de solutions candidates qui sont générées à chaque itération de l'algorithme,
- un nombre réel $\rho \in (0, 1)$, qui détermine le nombre de candidats N_b à chaque itération qui sont considérés comme les mieux classés,
- une séquence de paramètres de lissage $\{\alpha_t\}^n{}_{i=1}$ avec $\alpha_t \in (0, 1]$ pour tout t.

Nous choisissons la version lissée de l'algorithme CE car elle est spécialement efficace pour les problèmes d'optimisation impliquant des variables aléatoires discrètes [35] comme dans notre cas.
Plusieurs critères d'arrêt sont disponibles pour l'algorithme CE [36]. Nous optons, ici, pour le nombre maximum d'itérations.
La fonction objectif, comme elle été définie dans la section "Approche déterministe (DCA)", est la suivante

$$\max N_f = 2^{n-1} - \min_{w \in \mathbb{B}^n} \psi(f),$$

avec

$$\psi(f) = \max_{w \in \mathbb{B}^n} \left| \sum_{x \in \mathbb{B}^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in \mathbb{B}^n} a_{wx} \right|.$$

Nous appliquons DCA à l'ensemble de solutions candidates les mieux classées suivant le schéma CE-DCA.

Algorithme 8.6.1

1. *Initialiser p_0, N, ρ et $\{\alpha_t\}^n{}_{i=1}$ et calculer $Nb = N + 1 - [(1-\rho)N]$. Soit $t = 1$ (le compteur d'itérations).*
2. *Générer un ensemble de solutions candidates $\{X_t{}^k,\ k = 1, \cdots, N\}$, de la distribution $f(.; p_{t-1})$ et calculer la fonction d'évaluation $S(X_t{}^{(k)})$ pour tout k, et les mettre dans l'ordre croissant : $S_{(1)} \leq S_{(2)} \leq \cdots \leq S_{(N)}$. Soit $\mathcal{B}_t$ l'ensemble d'indices k correspondant aux fonctions d'évaluations $S_{\lceil (1-\rho)N \rceil}, ..., S_{(N)}$.*
3. *Appliquer la procédure DCA sur l'ensemble des éléments ayant les indices dans $\mathcal{B}_t$.*
4. *Pour $i = 1, \cdots, n$, calculer $w_{t,i} = \sum_{k \in \mathcal{B}_t} X_{t,i}{}^{(k)}/N_b$ avec $X_{t,i}{}^{(k)}$ est la i^{me} composante de $X^{(k)}{}_t$. Le vecteur paramètre est mis à jours selon la formule*

$$p_{t,i} = (1 - \alpha_t)\, p_{t-1,i} + \alpha_t w_{t,i}, i = 1 \cdots n.$$

5. *Si le critère d'arrêt est atteint alors stop, sinon soit $t = t + 1$ et recommencer à partir de l'étape 2.*

8.7 Expériences numériques

Pour l'implémentation de DCA, nous avons pris respectivement les valeurs 0 et 10^{-6} pour les paramètres b et ϵ.
Dans cette section, nous allons tester et comparer 7 algorithmes pour la génération de fonctions booléennes de haut degré de non-linéarité.

- **SA :** l'algorithme de recuit simulé en prenant comme fonction objectif $\Psi(f)$ dans (8.9). L'algorithme termine la recherche au bout de 300 cycles de température ou bien après 50 cycles consécutifs sans résultats acceptés. A chaque cycle de température, 400 pas sont considérés. Un facteur de température $\alpha = 0.9$ est utilisé dans la suite.
- **SADCA1** : le schéma SA-DCA.
- **SADCA2** : le schéma SA-DCA avec une fonction d'évaluation égale à celle utilisée par Clark [96].

$$cost\left(\hat{f}\right) = \sum_{\omega \in F^n} \left| \left| \hat{F}(\omega) \right| - 2^{\frac{n}{2}} \right|^R, \quad R = 3. \tag{8.12}$$

- **Two phase DCA** : le schéma SA-DCA où nous appliquons en première phase **SADCA1** et DCA est appliqué une deuxième fois en phase 2 à partir de la solution obtenue en phase 1.
- **SAHC :** c'est une approche à deux phases réalisée par Clark et Jacob [96]. Le recuit simulé est exécuté en première phase suivie par un algorithme de Hill-Climbing.
- **SADCAC :** l'algorithme SA-DCA1 en commençant par une fonction courbe.
- **CE-DCA :** l'algorithme d'entropie croisée combiné avec l'algorithme DCA. Les paramètres suivants ont été utilisés pour l'algorithme CE :

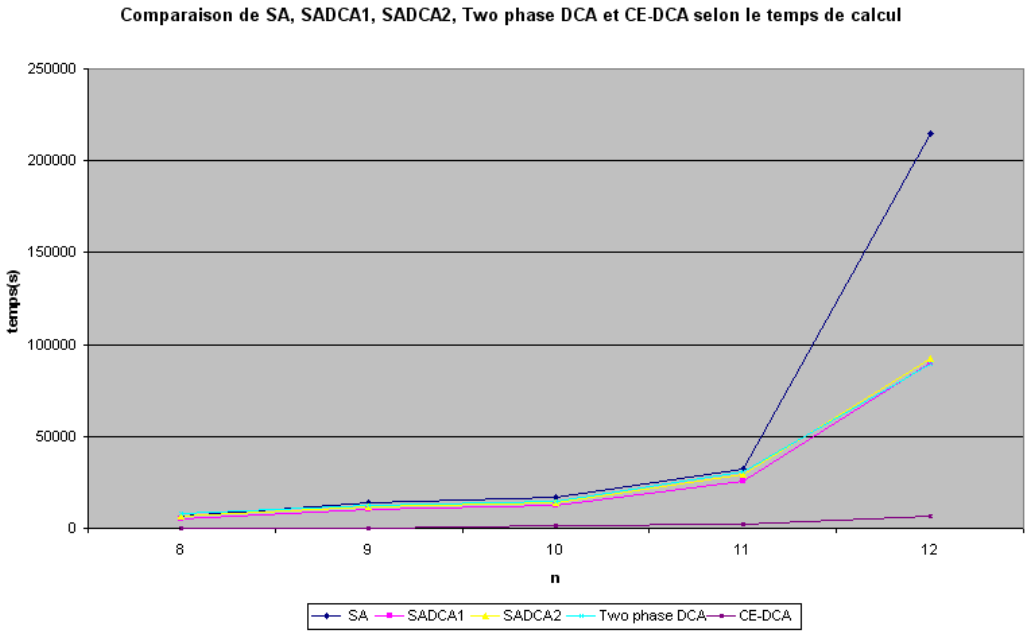
  - $\rho = 0,01$
  - N est déterminé par la technique expérimentale selon la valeur de n.
  - $\alpha = 0,4$
  - pour minimiser le temps de calcul, nous avons ajouté un autre critère d'arrêt. L'algorithme s'arrête si la valeur de $(S_{\lceil (1-\rho)N \rceil})$ ne change pas au bout de cinq itérations successives.

Technique	8		9	10	11	12
SA	NL	114	232	476	968	1984
	DCA-iter	-	-	-	-	-
	Temps CPU	1h 46min 10sec	3h 47min 40sec	4h 43min 17sec	8h 54min 15sec	59h 36min 15sec
SADCA1	NL	116	234	480	978	1986
	DCA-iter	9	8	9	5	6
	Temps CPU	1h 23min 20sec	2h 46min 10sec	3h 28min 19sec	7h 09min 30sec	24h 59min 19sec
SADCA2	NL	116	234	484	986	1984
	DCA-iter	5	8	6	4	6
	Temps CPU	1h 52min 10sec	3h 15min 48sec	3h 53min 01sec	8h 14min 45sec	25h 45min 10sec
Two phase DCA	NL	116	236	486	978	1986
	DCA-iter	4	2	3	2	3
	Temps CPU	2h 10min	3h 24min 16sec	4h 05min 00sec	8h 28min 14sec	24h 52min 26sec
SAHC	NL	116	236	484	984	1990
	DCA-iter	-	-	-	-	-
	Temps CPU	-	-	-	-	-
SADCAC	NL	116	-	488	-	1984
	DCA-iter	-	-	-	-	-
	Temps CPU	1min 7sec	-	1h 36min 4sec	-	24h 45min 20sec
CE-DCA	NL	116	236	476	972	1984
	DCA-iter	-	-	-	-	-
	Temps CPU	1min 20sec	4min 2sec	17min 22sec	39min 52sec	1h 51min 10sec

TABLE 8.1 – Comparaison de résultats des algorithmes SA, SADCA1, SADCA2, Two phase DCA, SAHC, SADCAC et CE-DCA pour (n=8, 9, 10, 11, 12)

L'observation de ces résultats montre que :

- Pour ($n \geq 8$) et ($n < 12$), nous avons considéré la moitié des cycles de température dans [96] mais SADCA2 arrive à donner la même valeur de non-linéarité. Il est, donc, clair que l'utilisation de DCA à chaque cycle de température a amélioré le temps de calcul de recuit simulé.
- Dans l'algorithme "Two phase DCA", seulement quelques itérations après le redémarrage de DCA ont suffit pour obtenir la meilleure solution.
- Toutes les fonctions booléennes obtenues sont équilibrées.
- Les résultats obtenus par SADCAC ont amélioré considérablement le temps de calcul.
- L'algorithme CE-DCA a bien amélioré le temps de calcul par rapport à SA-DCA1, SA-DCA2, "Two phase DCA" et SA. Le graphique (8.7) montre nettement le résultat.



- La qualité de résultats de CE-DCA peut être améliorée en utilisant d'autres techniques pour l'ajustement de paramètres [97].

8.8 Conclusion

Dans ce chapitre, nous avons utilisé, le premier modèle d'optimisation déterministe pour la construction des fonctions booléennes équilibrées de haut degré de non-linéarité. Le problème

a été reformulé en terme d'une programmation DC en utilisant un nouveau résultat sur la pénalité exacte, ce qui a permis de développer DCA pour sa résolution numérique. Nous avons présenté une approche hybride qui combine DCA et recuit simulé. Les résultats numériques ont montré que l'algorithme combiné est mieux que l'algorithme de recuit simulé exécuté seul. Cet algorithme a été efficace pour trouver un bon point de départ pour DCA. Par contre, l'algorithme "two phase DCA" reste le meilleur parmi tous les schémas présentés.
D'autres tentatives d'amélioration de la qualité de la solution ont combiné DCA à l'algorithme CE et commencé SA-DCA par une fonction courbe. Si cette dernière a amélioré nettement la solution, la première a diminué considérablement le temps de calcul.

Chapitre 9

Cryptanalyse d'un schéma d'identification basé sur le problème perceptron et le problème perceptron permuté

9.1 Introduction

Dans de nombreuses applications (courrier électronique, échange bancaire,..) il peut s'avérer nécessaire que l'expéditeur s'assure de l'identité du destinataire avant de débuter une transaction. C'est le problème de l'authentification. Dans la terminologie usuelle, les deux intervenants sont respectivement appelés le vérificateur et le prouveur et sont représentés par Alice et Bob.
De nos jours, les supports plastifiés au format carte de crédit sont de plus en plus utilisés comme moyen d'identification. Ainsi, lors de la mise au point d'un schéma d'identification, il est habituel de supposer que le prouveur dispose d'une faible puissance de calcul et d'une quantité limitée de mémoire. Selon le contexte, ces contraintes peuvent aussi s'appliquer au vérificateur. Lors de l'élaboration d'un schéma d'identification, hormis sa sécurité, trois paramètres essentiels doivent être pris en compte :

- le volume de quantités échangées (débit de transaction)
- la complexité des calculs pour le prouveur
- la capacité de stockage nécessaire au prouveur

Les problèmes de base concernant les méthodes d'identification actuelles proviennent du fait que le prouveur établit son identité en révélant un secret qu'il détient. Un vérificateur malhonnête peut, dans ce cas, usurper sans difficulté l'identité du prouveur.
En 1985, *S.Goldwasser*, *S.Micali* et *C. Rackoff* introduisent la notion de preuve interactive à

divulgation nulle (zero-knowledge). Ils démontrent que la quantité minimale d'information, qu'un individu doit dévoiler pour convaincre (avec une forte probabilité) son interlocuteur qu'il détient la solution s d'un problème difficile, peut être nulle. Après le schéma proposé en 1986 par *A. shamir* et *A.Fiat*, un autre schéma a été introduit par en 1995 par *David Pointcheval*.
Le schéma d'identification basé sur le problème perceptron a été introduit dans l'article "A new identification scheme based on the Perceptron Problem"[13] et a été considéré comme bien adapté à la carte à puce. En effet, ce schéma ne nécessite pas une puissance de calcul importante et ne peut pas être attaqué facilement.
Dans ce chapitre, nous allons commencer par définir le problème perceptron. Par la suite, Nous allons évoquer les différentes tentatives d'attaques réalisées sur le schéma d'identification basé sur ce problème. Ensuite, le modèle mathématique présenté dans [1] est utilisé afin d'implémenter un nouvel algorithme combinant le recuit simulé avec DCA.
Dans tout ce chapitre, soit $\mathcal{E} := \{-1, 1\}$. Une matrice (resp. vecteur) est dite $\mathcal{E}$-matrice (resp. $\mathcal{E}$ - vecteur) si toutes les composantes de la matrice (resp. vecteur) appartiennent à $\mathcal{E}$.

9.2 Définition du problème perceptron et du problème perceptron permuté

Définition 9.2.1
Le problème perceptron *(PP)*
Données : Une $\mathcal{E}$*-matrice* $A\,(m \times n)$
Problème : Trouver un vecteur $V \in \{-1, 1\}^n$ *vérifiant*
$\{(AV)_i \,|\, i = \{1, \cdots m\}\} \geq 0$

Définition 9.2.2
Le problème perceptron permuté*(PPP)*
Données :Une $\mathcal{E}$*-matrice* $A\,(m \times n)$*, un tableau* S *d'entiers non négatifs de taille* m*.*
Problème :Trouver un vecteur $V \in \{-1, 1\}^n$ *vérifiant*
$\{(AV)_i \,|\, i = \{1, \cdots, m\}\} = S$ *Le problème perceptron permuté est un problème NP-complet.*

Associé au problème d'apprentissage de lecture, ce problème est très connu dans le domaine d'intelligence artificielle.
Dans le domaine d'apprentissage, le perceptron peut être vu comme le type de réseau de neurones le plus simple. C'est un classifieur linéaire. Il a été inventé en 1957 par *Franck Rosenblat* au Cornell Aeronautical Laboratory, inspiré par la théorie cognitive de *Friedrich Hayek* et celle de *Donald Hebb*. Dans sa version simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Les entrées et la sortie sont booléennes. Le potentiel post-synaptique $\sum W_i e_i$ où W_i est le poids de l'entrée

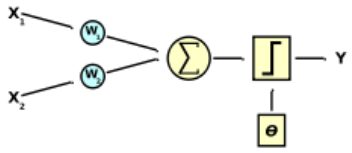


FIGURE 9.1 – Perceptron avec 2 entrées et une fonction d'activation à seuil

e_i. La fonction d'activation est la fonction de Heaviside (la fonction signe est parfois utilisée) $H(x) = \begin{Bmatrix} 0 & \text{si} & x < 0 \\ 1 & \text{si} & x \geq 0 \end{Bmatrix}$, avec $x = \sum W_i e_i - b$. Ici, b définit le seuil à dépasser pour que la sortie soit à 1. On trouve fréquemment une variante de ce modèle de base dans laquelle la sortie prend les valeurs -1 et 1 au lieu de 0 et 1.

9.3 Méthodes existantes

Pour mettre en valeur l'efficacité de son schéma, *Pointcheval* a procédé à des attaques sur (PP) en utilisant la technique de recuit simulé. Puisque toute solution de (PPP) est une solution pour (PP), le recuit simulé a été utilisé pour résoudre (PP) jusqu'à ce que une solution pour (PPP) est trouvée. En utilisant cette technique, aucune solution pour (PPP) telle que $(n > 71)$ ne pourra être trouvée même si on continue la recherche pour plusieurs heures. A la suite de ses attaques, *Poincheval* a proposé les matrices de taille $(m, n) = (m, m + 16)$ comme les matrices les plus sûres du point de vue de la sécurité. Il a aussi mentionné qu'en pratique, on peut prendre des matrices de tailles (101,117), (131,147) et (151,167)[13].
En 1999, *Knudsen* and *Meier* ont réussit à trouver une technique de recherche itérative basée sur le recuit simulée. Knudsen et Meier ont utilisé une nouvelle fonction objectif dans laquelle il utilise la fonction histogramme de S pénalisée.

$$E(X) = g_1 \sum_{i=1}^{m} (|(AX')_i| - (AX')) + g_2 \sum_{i=1}^{n} (|H_i - H'_i|).$$

$H(k) = card\{i, (AX)_i = k\}$, c'est à dire le nombre de $((AX)_i$ ayant la valeur $k)$.
En effet, la nouvelle technique arrive à trouver une solution même pour les instances de taille supérieure à (101,117) selon les auteurs mais ils ne précisent pas la borne supérieure obtenue. Malgré que le pourcentage de succès est loin d'être 100%, l'algorithme arrive souvent à déterminer une partie importante des composantes de la clé secrète. Les auteurs affirment que leur algorithme est très rapide (280 fois plus rapide que [13])
En 2002, J.A. Clark et J.L. Jacob ont remarqué que lors de la génération des données de PP, la négation de certaines lignes de la matrice peut rendre des valeurs positives de $(AX)_i$ négatives. On peut, aussi, se trouver piégé dans un minimum local en utilisant la fonction objectif

utilisée par PointCheval, Knudsen et Meier. Ils ont proposé la fonction objectif suivante comme solution pour éviter ces problèmes.

$$Cost(X') = g \sum_{i=1}^{m} (\max\{K - (AX')_i, 0\})^R.$$

K et R sont deux entiers positifs et g servira de poids lorsque la fonction objectif sera utilisée pour PPP.
Pour PPP, les auteurs ont proposé la fonction objectif suivante

$$Cost(X') = g \sum_{i=1}^{m} (\max\{K - (AX')_i, 0\})^R + g_2 \sum_{i=1}^{n} (|H_i - (H')_i|)^R.$$

Récemment, une nouvelle tentative d'attaque utilisant un modèle d'optimisation déterministe a été proposée au sein de LITA [98]. Les premiers résultats étaient encourageants. Notre travail consiste à améliorer les résultats obtenus dans [1] en augmentant la taille des instances.

9.4 Résolution du Problème Perceptron (PP)

Plusieurs attaques ont été effectuées sur le schéma d'identification basé sur le problème PP pour évaluer son efficacité. Une approche déterministe a été présentée dans [1]. Elle consiste à modéliser le problème comme un programme DC (Difference of Convex functions). Notre travail consiste à améliorer les résultats obtenus en utilisant une combinaison de l'algorithme de recuit simulé et l'algorithme DCA afin de tirer profit des avantages des deux algorithmes [99].

9.4.1 Formulation mathématique du problème

Nous considérons $\mathcal{P}$ l'ensemble des solutions du problème PP et $\mathcal{P}'$ son domaine continu relaxé.

$$\mathcal{P} := \{x \in \{-1,1\}^n : (Ax)_i \geq 0 \ \forall i = 1 \cdots m\}, \mathcal{P}' := \{x \in [-1,1]^n : (Ax)_i \geq 0 \ \forall i = 1 \cdots m\}.$$

Soit p la fonction définie dans $\mathbb{R}^n$ par :

$$p(x) := \sum_{i=1}^{n} (1 - x_i^2).$$

On a :

i) p est une fonction quadratique concave et $p(x) \geq 0,\ x \in \mathcal{P}'$.

ii) $\mathcal{P} = \{x \in \mathcal{P}' : p(x) = 0\}$.

Proposition 9.4.1
Si $\mathcal{P}$ est non vide alors PP est équivalent au problème suivant :

$$0 = \min \{p(x), x \in \mathcal{P}'\}. \tag{9.1}$$

Preuve :

D'après i), $\mathcal{P}'$ ne peut pas être vide, alors (9.1) admet une solution optimale.
Cette solution optimale doit être dans $\mathcal{P}$ d'après ii). Donc, l'ensemble de solutions optimales de (9.1) correspond à l'ensemble de solutions de PP. D'autre part, il est clair que si $\min \{p(x), x \in \mathcal{P}'\} = 0$ alors $\mathcal{P}$ n'est pas vide.

9.4.2 Résolution du problème (9.1) par l'algorithme DCA

9.4.2.1 Formulation DC

Soit $\chi_{\mathcal{P}'}$ la fonction indicatrice de $\mathcal{P}'$ définie comme suit :

$$\chi_{\mathcal{P}'} = \begin{cases} 0 \ si \ x \in \mathcal{P}', \\ +\infty \ sinon. \end{cases} \tag{9.2}$$

$\mathcal{P}'$ est un ensemble convexe. Le problème s'écrit ainsi

$$0 = \min \{F_{PP}(x) := \chi_{\mathcal{P}'}(x) - (-p(x)) : x \in \mathbb{R}^n\}. \tag{9.3}$$

Une décomposition naturelle de la fonction objectif $F_{PP}(x)$ serait donc

$$F_{PP}(x) = G_{PP}(x) - H_{PP}(x).$$

avec

$$G_{PP}(x) = \chi_{\mathcal{P}'}(x), \tag{9.4}$$

et

$$H_{PPP}(x) = -p(x) = -\sum_{i=1}^{n} (1 - x_i^2). \tag{9.5}$$

Les fonctions G_{PP} (x) et $H_{PP}(x)$ sont convexes, donc (PP) est un programme DC. Lors de la résolution de PP par DCA, nous devons construire deux suites $\{x^l\}$ et $\{y^l\}$ telles que :

$$y^l \in \partial H_{PP}(x^l),$$

$$x^{l+1} \in \partial G^*{}_{PP}(y^l).$$

9.4.2.2 Calculer y^l

La fonction $-p$ est différentiable et la valeur de son gradient est donnée par

$$\nabla(-p)(x) = 2x.$$

9.4.2.3 Calculer x^{l+1}

calculer $x^{l+1} \in \partial G^*{}_{PP}(y^l)$, revient à calculer

$$x^{l+1} = argmin\left\{G_{PP}(x) - \langle x - x^l, y^l\rangle : x \in \mathbb{R}^n\right\}. \tag{9.6}$$

Autrement dit, x^{l+1} est une solution optimale du problème linéaire suivant

$$\min\left\{-\langle x, y^l\rangle : x \in \mathcal{P}\right\}. \tag{9.7}$$

9.4.2.4 Schéma DCA : PP-DCA

Algorithme 9.4.1

Initialisation : choisir un vecteur $x^0 \in \mathbb{R}^n$, *soit* $\epsilon_{DCA} > 0$, $l \leftarrow 0$

Répéter

1. *Calculer* $y^l = 2x^l$,
2. *Résoudre le problème ci-dessous pour trouver* x^{l+1},

$$\min\left\{-\langle x, y^l\rangle : x \in \mathcal{P}\right\}$$

3. $l \leftarrow l + 1$,

Jusqu'à $\left(\left\|x^l - x^{l-1}\right\| \leq \epsilon_{DCA}\right)$ *ou* $\left(p(x^l) = 0\right)$.

Les propriétés de convergence de **PP-DCA** se trouvent dans le théorème suivant

Théorème 9.4.1

i. ***PP-DCA*** *génère une suite* $\left\{x^l\right\}$ *appartenant à l'ensemble des sommets de* $\mathcal{P}'$ *noté* $V(\mathcal{P}')$ *telle que la suite* $\left\{p\left(x^l\right)\right\}$ *soit décroissante.*

ii. *La suite* $\left\{x^l\right\}$ *converge vers* $x^* \in V(\mathcal{P}')$ *après un nombre fini d'itérations.* x^* *est un point critique du problème. En plus, si* ${x^*}_i \neq 0$ *(resp.* ${x^*}_i \in \{-1, 1\}$*) pour tout* $i \in \{1 \cdots n\}$, *alors* x^* *est une solution locale (resp. globale)*

9.4.2.5 Procédure de redémarrage de DCA

Nous avons utilisé la procédure de redémarrage décrite dans [1]. La procédure consiste à faire le traitement suivant :

Soit i le premier indice pour lequel x_i n'est pas dans $\{-1, 1\}$
- Si $0 \leq x_i < 1$ alors $x_i \leftarrow 1$
- Si $0 > x_i > -1$ alors $x_i \leftarrow -1$

Le nouveau schéma de DCA (**RDCA**) est, alors

Algorithme 9.4.2
Initialisation : choisir un vecteur $x^0 \in \mathbb{R}^n$. *Soit* $\epsilon_{DCA} > 0$, $l \leftarrow 0$
Répéter

1. *Calculer* $y^l = 2x^l$;
2. *Résoudre le problème ci-dessous pour trouver* x^{l+1}

$$\min \left\{ \langle -x, y^l \rangle : x \in \mathcal{P}' \right\}.$$

3. *Soit* i *le premier indice pour lequel* x_i *n'est pas dans* $\{-1, 1\}$,
   - *Si* $0 \leq x_i < 1$ *alors* $x_i \leftarrow 1$
   - *Si* $0 > x_i > -1$ *alors* $x_i \leftarrow -1$
4. $l \leftarrow l + 1$

Jusqu'à $\left(\left\|x^l - x^{l-1}\right\| \leq \epsilon_{DCA}\right)$ *ou* $\left(p(x^l) = 0\right)$.

9.4.2.6 Choix d'un bon point initial pour DCA

Le choix d'un bon point initial pour DCA est une tâche difficile et cette question reste ouverte. Cependant un début de réponse semble être l'adjonction à DCA d'une procédure pour calculer un bon point initial. Une telle procédure pourrait être un autre algorithme. Dans notre cas nous avons utilisé un algorithme génétique qui va nous générer une population de solutions réalisables. Parmi ces solutions, la meilleure sera choisie comme point de démarrage de DCA.
Nous allons maintenant définir les éléments principaux de notre algorithme génétique, à savoir : le codage, la fonction fitness, la population initiale et les opérateurs génétiques (croisement et mutation).
Représentation : Pour les algorithmes génétiques, un des facteurs les plus importants, si ce n'est le plus important, est la façon dont sont codées les solutions (ce qui a été nommé ici les chromosomes), c'est-à-dire les structures de données qui coderont les gènes. *Le principe des alphabets minimaux* annoncé par GOLDBERG dans son livre et définit ainsi :
l'utilisateur doit choisir le plus petit alphabet qui permette une expression naturelle du problème

Un codage qui s'impose naturellement est, alors, le codage à caractères multiples qui consiste à considérer les chromosomes comme étant des chaines de $\{-1, 1\}$ représentant le vecteur x de taille n que nous cherchons.

Population initiale : Nous avons testé plusieurs valeurs. En effet, si celle-ci est trop petite, il faut beaucoup plus de générations pour atteindre le maximum, car il n'y a pas assez de variété dans la population initiale. Seules les mutations finissent par permettre à l'algorithme de converger. Nous avons, donc, opté pour la valeur : 30.

Fonction d'évaluation : La fonction d'adaptation utilisée est la suivante :

$$F(x) = \sum_{i=1}^{m} |(AV)_i| - (AV)_i.$$

Sélection : La sélection utilisée dans notre algorithme génétique est la sélection uniforme. La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité $\frac{1}{p}$ d'être sélectionné, où p est le nombre total d'individus dans la population.

Croisement : Nous appliquons un croisement en un point classique. La probabilité de croisement est fixée : $p_c = 0.7$.
Mutation : la mutation vise à modifier de façon aléatoire une partie de la population. Ici, le principe est d'inverser aléatoirement la valeur de l'un des gènes des individus de la population concernée : $p_m = 0.5$.
Condition de terminaison : Comme dans la plupart des algorithmes génétiques, nous utilisons un nombre maximal de générations pour la condition d'arrêt de l'algorithme. Nous fixons le nombre d'itérations à 100.
Pseudo code de l'algorithme génétique utilisé

Algorithme 9.4.3
Initialisation
Générer la population initiale P_t contenant N individus.
Evaluer la population P_t.
Pour nbIteration de 1 à nbIterMax

- *Sélectionner deux individus i_1 et i_2 dans P_t.*
- *Croiser i_1 et i_2 avec une probabilité $p_c = 0.7$.*
- *Appliquer la mutation avec une probabilité $p_m = 0.5$.*
- *Insérer les nouveaux individus dans P_{t+1}*
- $P_t \leftarrow P_{t+1}$
- *Evaluer la population P_t.*

9.4.3 Résultats numériques

Pour l'implémentation de nos algorithmes et pour les applications cryptographiques, on a besoin des instances du problème pour lesquelles on connaît une solution. Nous générons aléatoirement, un vecteur V de taille n, qui va être la solution de l'instance considérée. D'autre part, nous générons une matrice aléatoire A(m,n) qui va être modifiée de la façon suivante :
Pour $i = 1 \cdots m$
- Si $(AV)_i < 0$, la i^{me} ligne de A est multiplié par -1.
- Si $(AV)_i > 0$, la i^{me} ligne de A est gardée.

Dans cette section, nous allons comparer nos algorithmes **RDCA** et **RDCA-GA** (exécutés en considérant 100 instances de problème PP) avec la méthode de recuit simulé **RC** présentée par knudsen et Meier.
RDCA : DCA avec une procédure de redémarrage et solution initiale générée aléatoirement, les critères d'arrêt considérés sont : $p(x) = 0$, c'est à dire que la solution est optimale ou bien que le temps d'exécution dépasse le temps d'exécution de recuit simulé.
RDCA-GA : l'algorithme RDCA avec un point de départ généré par un algorithme génétique.
RC : l'algorithme de recuit simulé utilisant la fonction objectif dans [13].
Les algorithmes sont comparés selon le pourcentage de réussite et le temps de calcul.
Temps de calcul : le temps global de calcul divisé par le nombre d'instances considérées (100).
Le pourcentage de réussite : le nombre de tests qui arrivent à trouver le vecteur secret x divisé par le nombre global de tests.

Les résultats numériques montrent que :

- **RDCA** et **RDCA-GA** sont plus rapides que le recuit simulé **RC**. Le graphique (**Comparaison des algorithmes suivant le temps de calcul**) réalisé sur les premières valeurs de (m,n) confirme bien le résultat.
- L'algorithme combiné **RDCA-GA** arrive à attaquer toutes les instances de PP pour $m \geq 501$ et le pourcentage de réussite est de 97.50% pour $m \leq 4001$.
- Les algorithmes **RDCA-GA** et **RDCA** arrivent à attaquer PP même pour les problèmes ayant une taille importante en un temps raisonnable. L'algorithme **RC**, lui, tourne sans succès (pendant des jours) à partir des instances ayant $m > 4001$.
- Il s'est révélé intéressant de tester les algorithmes proposés pour des tailles plus importantes pour arriver à déterminer la taille à partir de laquelle nous ne pouvons plus trouver x.

m	n	RDCA		RDCA-GA		RC	
		Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)
101	117	0.293	100	0.383	100	3.437	100
121	137	0.312	100	0.686	100	4.578	100
131	147	0.403	96	0.785	100	5.781	81
151	167	0.582	100	0.853	100	6.265	100
171	187	0.718	100	0.998	100	6.719	100
201	217	0.984	100	1.568	100	55.95	100
301	317	1.825	100	2.943	100	147.781	100
401	417	6.230	96	8.750	100	199.98	97
501	517	5.546	100	9.718	100	305.47	94
801	817	13.343	96	16.953	98	1911.53	92
1001	1017	22.031	97	41.200	99	3208	94
2001	2017	115.85	75	162.06	91	3651	74
3001	3017	191.75	89	286.93	96	4085	80
4001	4017	588.343	81	750.265	81	4575	81
La moyenne		67.73	95	91.72	97.50	1297.62	92.36
5001	5017	1816.41	77	2153	78	-	-
6001	6017	5916.25	65	6409.06	82	-	-
4000	8017	8389.18	86	9546.24	88	-	-

TABLE 9.1 – Comparaison entre RDCA,RDCA-GA,RC sur 100 instances de PP.

9.5 Résolution du problème perceptron permuté (PPP)

9.5.1 Formulation mathématique du problème

Le problème PPP dérive de PP et il diffère de ce dernier par la contrainte suivante, qui fera, en fait, toute la difficulté de PPP.

$$\{\{(Ax)_i | i = 1 \cdots m\}\} = S.$$

Si nous considérons l'exemple de problème PPP suivant :

$$\begin{pmatrix} & \mathbf{A} & & & \\ 1 & -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} \mathbf{S} \\ 3 \\ 1 \\ 1 \\ 5 \end{pmatrix}$$

H est l'histogramme de valeurs de S : H(S)=(h(1),h(3),h(5))=(2,1,1) (voir la figure 9.2)

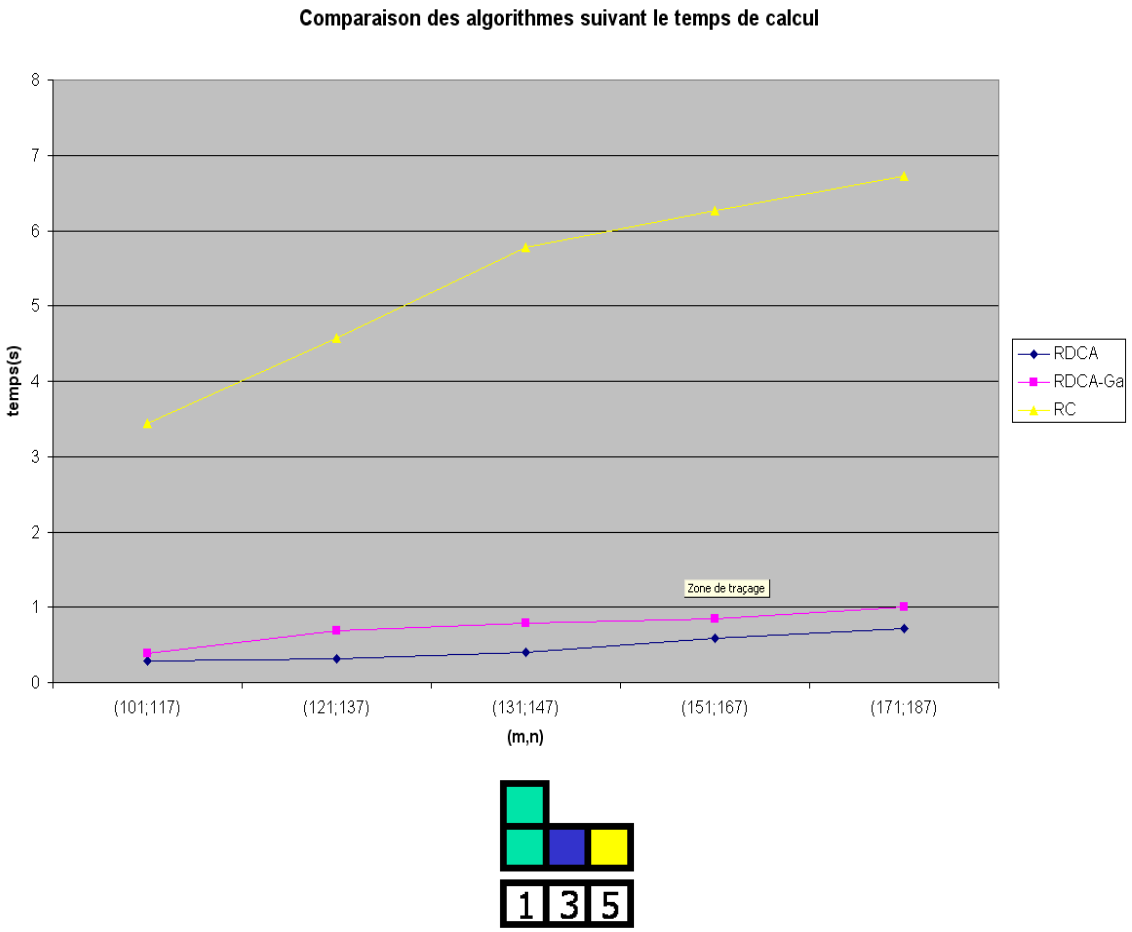


FIGURE 9.2 – Histogramme de S

Le problème a été modélisé dans [1] de la façon suivante :
Le multiset S est désormais représenté par deux vecteurs :

- S^* : contient l'ensemble des éléments distincts de S. Nous notons p son cardinal.
- C : contient le nombre d'occurrences de chaque élément de S^*.

Afin d'exprimer la relation entre $\{(Ax)_i | i = 1 \cdots m\}$ et S, une nouvelle variable y a été introduite. $y(m, p)$ est une matrice telle que

$$y_{ij} = \begin{cases} 1 \; si \; (Ax)_i \; = \; S^*[j], \\ 0 \; sinon. \end{cases} \tag{9.8}$$

Autrement dit, dans le cas idéal $(Ax)_i \; = \; {S^*}_j$ sinon la valeur de $(Ax)_i$ doit être comprise entre le (S_{max}) et le minimum (S_{min}) de tableau S.. Pour exprimer cela, la contrainte suivante

est ajoutée à notre programme

$$\begin{cases} A_i x \leq s_{max} - (s_{max} - {s^*}_j) \forall i = 1 \cdots m, j = 1 \cdots p \\ A_i x \geq s_{min} + ({s^*}_j - s_{min}) \forall i = 1 \cdots m, j = 1 \cdots p \end{cases} \tag{9.9}$$

$\{\{(Ax)_i | i = 1..m\} = S\}$ exprime le fait que chaque élément $(Ax)_i$ doit correspondre à un élément dans S^*. C'est pour cela que chaque ligne de la matrice y doit avoir un et un seul élément égal à 1.

$$\sum_{j=1}^{p} y_{i,j} = 1 \forall i = 1 \cdots m. \tag{9.10}$$

Comme pour chaque ${s^*}_j$, nous devons avoir exactement c_j éléments dans $(Ax)_i$ tels que $(Ax)_i = {s^*}_j$. La contrainte suivante a été ajoutée

$$\sum_{i=1}^{m} y_{i,j} = c_j \forall j = 1 \cdots p. \tag{9.11}$$

Le domaine de solutions de PPP K_{PPP} est alors défini par

$$K_{PPP} = \left\{ \begin{array}{l} \{x, y\} \in [-1, 1]^n \times \{0, 1\}^{m \times p} \\ \sum_{j=1}^{p} y_{i,j} = 1 \forall i = 1 \cdots m \\ \sum_{i=1}^{m} y_{i,j} = c_j \forall j = 1 \cdots p \\ \sum_{j=1}^{n} a_{ij} x_j \leq s_{max} + (s_i - s_{max}) y_{ij} \forall i = 1 \cdots m \forall j = 1 \cdots p \\ \sum_{j=1}^{n} a_{ij} x_j \geq s_{min} + (s_i - s_{min}) y_{ij} \forall i = 1 \cdots m \forall j = 1 \cdots p \end{array} \right\}. \tag{9.12}$$

De manière analogue au problème PP, on peut facilement démontrer que PPP est équivalent au problème d'optimisation suivant :

$$PPP : \min \left\{ \sum_{i=1}^{n} \min \{(1 + x_i), (1 - x_i)\} + \sum_{i=1}^{m} \sum_{i=1}^{p} \min \{(y_{ij}, (1 - y_{ij})\} : \{x, y\} \in K_{PPP} \right\}$$

où

$$K_{PPP} = \left\{ \begin{array}{l} \{x, y\} \in [-1, 1]^n \times [0, 1]^{m \times p} \\ \sum\limits_{j=1}^{p} y_{i,j} = 1 \forall i = 1 \cdots m \\ \sum\limits_{i=1}^{m} y_{i,j} = c_j \forall j = 1 \cdots p \\ \sum\limits_{j=1}^{n} a_{ij} x_j \leq s_{max} + (s_i - s_{max}) y_{ij} \forall i = 1 \cdots m \forall j = 1 \cdots p \\ \sum\limits_{j=1}^{n} a_{ij} x_j \geq s_{min} + (s_i - s_{min}) y_{ij} \forall i = 1 \cdots m \forall j = 1 \cdots p \\ \end{array} \right\}. \tag{9.13}$$

9.5.2 Résolution de PPP par DCA

9.5.2.1 Formulation DCA

La résolution de PPP par DCA nécessite la réécriture de sa fonction objectif comme étant la différence de deux fonctions convexes. Ainsi, en considérant $\chi_{K_{PPP}}$, PPP est équivalent au

$$(PPP :) \min \left\{ \chi_{K_{PPP}}(x, y) - H_{PPP}(x, y) : (x, y) \in \mathbb{R}^n \times \mathbb{R}^{m \times p} \right\} \tag{9.14}$$

avec

$$G_{PPP}(x, y) \; = \; \chi_{K_{PPP}}(x, y) \tag{9.15}$$

et

$$H_{PPP}(x, y) = -\sum_{i=1}^{n} \min \{(1 + x_i), (1 - x_i)\} - \sum_{i=1}^{m} \sum_{j=1}^{p} \min \{(y_{ij}, (1 - y_{ij})\} \tag{9.16}$$

G_{PPP} et H_{PPP} sont deux fonctions convexes.

La résolution de PPP par DCA consiste à calculer à chaque itération les deux suites $\left\{(x, y)^l\right\}$ et $\left\{(u, v)^l\right\}$ telles que

$$\begin{array}{c} \left(u^l, v^l\right) \in \partial H_{PPP}\left(x^l, y^l\right), \\ \left(x^{l+1}, y^{l+1}\right) \in \partial G^*_{PPP}\left(u^l, v^l\right). \end{array}$$

9.5.2.2 Calculer $\left(u^l, v^l\right) \in \partial H_{PPP}\left(x^l, y^l\right)$

$$u_i^l = \left\{ \begin{array}{l} -1 \; si \; x_i^l \leq 0 \; pour \; i = 1 \cdots n, \\ \; 1 \; sinon. \end{array} \right. \tag{9.17}$$

et

$$v_{i,j}^{l} = \begin{cases} -1 \ si \ y_{i,j}^{l} \leq 0.5 \ pour \ i = 1 \cdots m, \ j = 1 \cdots p, \\ \ \ 1 \ sinon. \end{cases} \tag{9.18}$$

9.5.2.3 Calculer$(x^{l+1}, y^{l+1}) \in \partial G^*{}_{PPP}(u^l, v^l)$

Cette condition est équivalente à :

$$\left(x^{l+1}, y^{l+1}\right) \ \in \ arg \ min \left\{ -\langle \left(u^l, v^l\right), (x, y) \rangle : (x, y) \in K_{PPP} \right\}. \tag{9.19}$$

9.5.2.4 Le schéma DCA pour la résolution de (PPP) : PPP-DCA

L'algorithme DC.
Initialisation : Soit $x^0 \in \mathbb{R}^n$, calculer y_0 selon (9.8), l=0 et ϵ_{DCA} est choisi suffisamment petit et de valeur positive.
Répéter

1. Calculer à chaque itération, $\left(u^l, v^l\right)$ par la formule (9.17) et (9.18) ;
2. Calculer $\left(x^{l+1}, y^{l+1}\right)$ en résolvant $arg \ min \left\{ -\langle \left(u^l, v^l\right), (x, y) \rangle : (x, y) \in K_{PPP} \right\}$;
3. $l := l + 1$.

Jusqu'à $\left\| (x^l, y^l) - (x^{l-1}, y^{l-1}) \right\| \leq \epsilon_{DCA} \left(\left\| \left(x^l, y^l\right) \right\| + 1 \right)$ ou $F_{PPP}\left(x^l, y^l\right) = 0$

Théorème 9.5.1
*(Les propriétés de convergence de l' algorithme **PPP-DCA**)*

*(i) **DCA** génère une séquence $\{\left(x^l, y^l\right)\}$ contenue dans $V(K_{PPP})$ telle que la séquence $\{F_{PPP}\left(x^l, y^l\right)\}$ est décroissante.*

(ii) La séquence $\{(x^l, y^l)\}$ converge à $\{(x^*, y^*)\} \in V(K_{PPP})$ après un nombre fini d'itérations. Le point $\{(x^*, y^*)\}$ est un point critique du problème (PPP). En plus si ${x^*}_k \neq 0$ et ${y^*}_{i,j} \neq 0.5$ pour tout $k \in \{1 \cdots n\}$, $i = \{1 \cdots m\}, j = \{1 \cdots p\}$ alors $\{(x^*, y^*)\}$ est une solution locale de (PPP).

9.5.2.5 Procédure de redémarrage de DCA

La procédure de redémarrage consiste à modifier les composantes de x qui ne sont pas dans $\{-1, 1\}$ de la façon suivante :

- Si $0 \leq x_i < 1$ alors $x_i = 1$
- Si $0 > x_i > -1$ alors $x_i = -1$

La modification de composantes de y qui ne sont pas dans $\{0, 1\}$ de la façon suivante

- Si $0.5 \leq y_i$ alors $y_i = 1$
- Si $0.5 \geq y_i$ alors $y_i = 0$

9.5.2.6 Choix du point initial de DCA

La recherche d'un bon point initial est toujours une question importante dans la résolution d'un programme DC par DCA. Elle dépend de la structure du problème. Dans cette partie, nous allons étudier différentes méthodes pour chercher un point initial.

Le vecteur majorité

Le vecteur majorité M est définit comme suit :

$$\begin{cases} M_j = 1 \ si \ |\{i|A_{i,j} = 1\}| > \dfrac{m}{2} \\ M_j = -1 \ sinon \end{cases}$$

Ce vecteur, proposé par D. Pointcheval, est considéré comme étant la première tentative d'attaque aux problèmes PP et PPP. En effet, D. Pointcheval a montré que le cas où m et n sont impairs, la corrélation entre vecteur majorité M et vecteur secret x est très importante.

$$\{j|M_j = x, j = 1 \cdots n\} \approx 0.8n$$

Algorithme génétique

Nous appliquons le même algorithme génétique pour la génération de point initial de PP.

Algorithme PP

Nous commençons par une des solutions obtenue lors de la résolution de PP et nous essayons de l'améliorer pour qu'elle devienne une solution de PPP.

9.5.3 Résultats numériques

Les données de PPP sont générées de manière analogue à PP. Une fois A et V sont déterminés, nous calculons S. Dans cette section, Nous allons comparer trois algorithmes avec l'algorithme de recuit simulé utilisant la fonction objectif présentée dans [100] :

- **PPP-GADCA :** l'algorithme DCA pour PPP en prenant comme point initial un vecteur généré par GA.
- **PPP-DCA :** l'algorithme DCA pour PPP en prenant comme point initial un vecteur généré aléatoirement.
- **PPP-SADCA :** l'algorithme de recuit simulé intégrant une procédure DCA exécutée à chaque pas de l'algorithme implémenté selon le schéma SA-DCA.
- **PPP-SA :** l'algorithme utilisant la fonction objectif dans [100].

Pour **PPP-SADCA**, la recherche est terminée au bout de 300 cycles de température ou bien après 50 cycles consécutifs sans résultats acceptés. A chaque cycle de température, 400 pas sont considérés. Le facteur de température considéré est $\alpha = 0.95$.
Pour l'implémentation de DCA, nous avons pris $\epsilon = 10^{-6}$

Comme pour PP, les algorithmes sont exécutés pour 100 instances et ils sont comparés suivant le pourcentage de réussite et le temps de calcul.

m	n	PPP-GADCA		PPP-DCA		PPP-SADCA		PPP-SA	
		Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)
73	73	1.766	80	1.031	68	28.969	66	45.217	64
81	81	2.328	72	1.251	70	56.687	67	73,437	58
101	101	3.547	60	2.092	62	79.546	42	81,437	32
101	81	5.468	40	2.234	32	106.984	36	113,032	35
121	81	8.375	46	2.986	44	173.98	52	194,004	52
201	217	9.656	54	3.094	51	183.15	52	211.63	49
301	317	13.656	49	4.953	49	373.65	52	399.21	40
401	417	20.219	47	7.156	32	438.625	20	520.78	10

TABLE 9.2 – *Comparaison entre PPP-DCA,PPP-GADCA,PPP-SADCA et PPP-SA.*

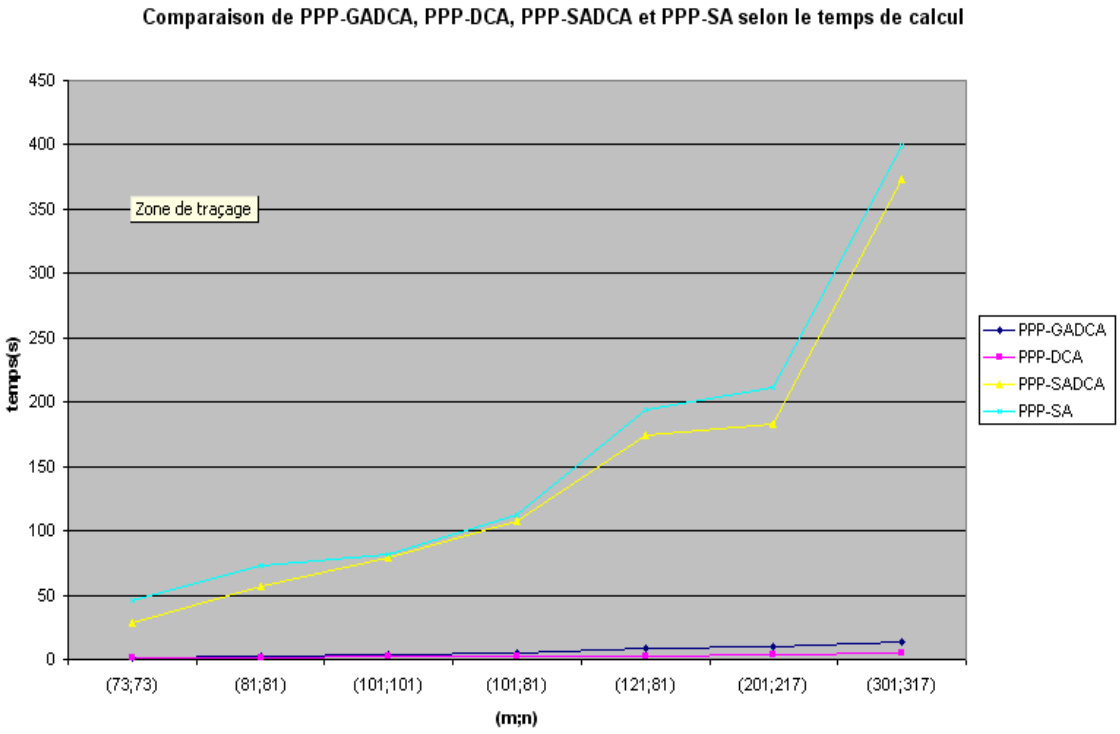
Les résultats numériques montrent que :

- **PPP-SADCA** a amélioré la qualité de la solution. La procédure DCA accélère le processus de SA et atténue l'effet de hasard en améliorant la qualité de la solution candidate.
- **PPP-GADCA** fournit les meilleurs résultats (voir graphique : **Comparaison de PPP-GADCA, PPP-DCA, PPP-SADCA, PPP-SA selon le temps de calcul**). GA est exécuté une seule fois pour trouver le bon point de départ. Une fois trouvé, c'est DCA qui prend le relais (DCA atteint souvent l'optimalité si on commence par un point de départ qui s'approche de celui là : visiblement GA arrive à réaliser cela).

Afin d'améliorer le taux de réussite de **PPP-DCA**, **PPP-SADCA** et **PPP-SA** pour $m \geq 201$ et $n \geq 217$, nous avons réalisé une série de tests en commençant par un point de départ une solution de problème PP.

Les résultats numériques montrent que :

– En commençant par une solution de problème PP, nous améliorons non seulement le temps de calcul mais aussi la qualité de solutions par rapport à la première série de tests. (PP donne de meilleurs résultats lorsqu'il commence par un bon point de départ)



m	n	PPP-DCA		PPP-SADCA		PPP-SA	
		Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)
201	217	5.609	32	142.781	75	153.423	30
301	317	7.89	49	216.547	51	271.021	34
401	417	14.077	52	118.241	51	420.78	45

TABLE 9.3 – *Comparaison entre PPP-DCA,PPP-SADCA,PPP-SA en les commençant par une solution de PP.*

9.6 Conclusion

Dans ce chapitre, nous avons étudié le Problème Perceptron et le Problème Perceptron Permuté. Ces deux problèmes sont combinatoires NP-difficiles et de très grande dimension en pratique.
L'étude porte sur une tentative d'attaque afin d'évaluer la sureté de schémas d'identification basé dessus. Les premières attaques sur ce schéma ont utilisé la technique de recuit simulé.
Notre contribution consiste à utiliser des modèles d'optimisation déterministe de deux problèmes pour implémenter plusieurs versions de l'algorithme DCA en modifiant à chaque fois la technique de génération de son point initial. Nous avons aussi intégrer la procédure DCA

dans l'algorithme de recuit simulé afin d'améliorer ses performances. Les résultats obtenus sont plutôt encourageants.

Conclusion et Perspectives

Dans cette thèse, nous avons étudié quatre problèmes importants de la cryptologie moderne s'intéressant aux deux aspects : cryptographie et cryptanalyse.
Pour les fonctions courbes, qui sont très importantes dans la cryptographie moderne, nous avons pu bâtir un modèle d'optimisation déterministe (combinatoire) puis le transformer en un programme DC. L'implémentation de DCA tout seul n'a pas donné le résultat souhaité. Nous avons donc combiné le recuit simulé avec une procédure DCA et les résultats étaient satisfaisants.
Dans les prochains travaux, d'autres modèles peuvent être mis en place en considérant d'autres propriétés de fonctions courbes.
La construction des fonctions équilibrées de haut degré de non-linéarité a été réalisée grâce à un algorithme hybride à partir duquel nous avons implémenté plusieurs schémas d'hybridation entre DCA et le recuit simulé.
Cet algorithme nous a permis de profiter des avantages de deux algorithmes : recuit simulé et DCA.
Les techniques utilisées dans notre travail peuvent être utilisées pour trouver des fonctions booléennes qui vérifient d'autres critères ou bien des fonctions booléennes combinant plusieurs critères.

Concernant les deux problèmes PP et PPP en cryptanalyse, qui sont très difficiles et attirent l'attention particulière de nombreux chercheurs dans ce domaine, nous avons utilisé la formulation du problème PP sous la forme de la minimisation d'une forme quadratique concave. Nous avons utilisé DCA, qui a été appliqué avec grand succès aux programmations quadratiques non convexes dans les travaux précédents pour la résolution de problème. Nous avons continué dans la même logique d'hybridation mais cette fois-ci avec l'algorithme génétique afin de trouver un bon point de départ pour DCA, vu que le problème de point initial reste toujours une question ouverte pour DCA. Les résultats obtenus étaient intéressants parce que nous sommes arrivés à traiter des instances de problème PP de taille beaucoup plus grande que celles résolues par les méthodes existantes.
Pour le problème PPP, nous avons utilisé le modèle établi dans [1]. Notre contribution a consisté à :

- trouver un "bon" point de départ pour DCA et ceci a été réalisé par l'algorithme génétique et par le résultat obtenu par PP
- profiter des avantages de deux algorithmes, recuit simulé de DCA, pour implémenter un nouvel algorithme hybride.

Les résultats obtenus sont encourageants.

Bibliographie

[1] Le Hoai Minh. *Modélisation et Optimisation non convexe basées sur la programmation DC et DCA pour la résolution de certaines classes des problèmes en Fouille de Données et Cryptologie.* PhD thesis, Université Paul Verlaine de Metz, 2007.

[2] Wild Beightler, Phillips. *Foundations of optimization.* 2nd ed.).Englewood Cliffs, 1979.

[3] Le Thi Hoai An. *Analyse numérique des algorithmes de l'optimization D.C : Approches locale et globale.* PhD thesis, Université de Rouen, 1994.

[4] Pham Dinh Tao Le Thi Hoai An, editor. *DC Programming : Theory, Algorithms and Applications.The State of the Art(28 pages)*, Valbonne-Sophia Antipolis, France, 2002. The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02). (containing the refereed contributed papers).

[5] Pham Dinh Tao Le Thi Hoai An. Convex analysis approach to dc programming : Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1), 1997. dedicated to Professor Hoang Tuy on the occasion of his 70th birthday.

[6] Pham Dinh Tao Le Thi Hoai An. Dc optimization algorithm for solving the trust region problem. *SIAM Journal on Optimization*, 8(2) :476–505, 1998.

[7] Pham Dinh Tao Le Thi Hoai An. Large scale molecular optimization from distance matrices by a dc optimization approach. *SIAM Journal on Optimization*, 14(1) :77–116, 2003.

[8] Pham Dinh Tao Le Thi Hoai An. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133 :23–46, 2005.

[9] Pham Dinh Tao Le Thi Hoai An. A global continuous approach based on dca and bb for solving mixed 0-1 linearly constrained polyhedral convex minimization problems, *The 6-th International Conference on Optimization : Techniques and Applications (*ICOTA6*).*

[10] Pham Dinh Tao Le Thi Hoai An. A continuous approach for globally solving mixed integer programming. *Sixth SIAM Conference on Optimization*, 1999.

[11] Le Thi Hoai An. Dc programming. Internet address : http ://lita.sciences.univ-metz.fr/lethi/DCA.html, 2006.

[12] François Bertrand Akoa. *Approches de points intérieurs et de la programmation DC en optimisation non convexe.* PhD thesis, Institut national des sciences appliquées de Rouen, 2005.

[13] David PoinCheval. A new identification scheme based on the perceptron problem. Eurocrypt'95, 1995.

[14] R.T. Rockafellar. *Convex Analysis.* 28. Princeton Mathematics-Princeton University Press,, Princeton, New Jersey, 1993.

[15] C. Lemarechal J.B. Hiriart-Urruty. Convex analysis and minimization algorithms. *Grundlehren der Mathematischen Wissenschaften-Springer-Verlag*, pages 305–306.

[16] R. Temam L. Ekeland. *Analyse Convexe et Problèmes Variationnels.* Dunod, Paris, 1974.

[17] J.B. Hiriart-Urruty. Generalized differentiability, dualityand optimiation for problems dealing with differences of convex functions. *Lecture Notes in Economics en MAth.Systems*, pages 37–70, 1985.

[18] J.B. Hiriart-Urruty. From convex optimization to nonconvex optimization. part I : Necessary and sufficient conditions for global optimality. *Nonsmooth Optimization and Related Topics, Ettore Majorana International Sciences,Series 43.*

[19] El-Ghazali Talbi. *Metaheuristics : from design to implementation.* John wiley and Sons Hobboken, New Jersey, 2009.

[20] David.E.Gold berg. *Algorithmes génétiques.* Editions Addisson-Wesley France, 1991.

[21] P. Siarry et E. Taillard J. Dréo, A. Pétrowski. *Metaheuristics for Hard Optimization.* Eyrolles, Paris, 2006.

[22] Jean-Michel Renders. *Algorithmes génétiques et réseaux de neurones.* Hermes, 1995.

[23] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[24] R. Cerf. *Une théorie asymptotique des algorithmes génétiques.* PhD thesis, Université de Montpelier II, 1994.

[25] Gelatt.C.D et Vecchi.M.P Kirkpatrick.S. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[26] V. Cerny. A thermodynamical approach to the travelling salesman problem : an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45.

[27] Siarry.P et Rahoual.M. Méthodes et pratiques de l'ingénieur. In *Réseaux informatiques : Conception et Optimisation.*

[28] Berthiau Gérard. *La méthode du recuit simulé pour la conception des circuits électroniques : adaptation et comparaison avec d'autres méthodes d'optimisation.* PhD thesis, Ecole centrale des arts et manufactures, Chatenay-Malabry, FRANCE, 1994.

[29] Fieguth.P et Lee.L.J Jamieson.M. Parametric contour estimation by simulated annealing. *ICIP03*, 3 :449–452, 2003.

[30] D. Naddef et G. Mounié O. Briant. Greedy approach and multi-criteria simulated annealing for the car sequencing problem. *European Journal of Operational Research.*

[31] E. H. L. et Laarhoven.V Aarts. Statistical cooling : A general approach to combinatorial optimization problems. *PHILIPS J. RES*, 40(4) :193–226.

[32] http ://www.proba.jussieu.fr. Théorie de doeblin "globale". `http://www.proba.jussieu.fr/cours/dea/telehtml/node37.html`, 2009.

[33] Gilles Le Beau. Introduction à l'analyse de l'algorithme de metropolis. `http://math.unice.fr/~sdescomb/MOAD/CoursLebeau.pdf`, 2009.

[34] R.Y.Rubinstein. Optimization of computer simulation models with rare events. *European Jornal of Operation Research*, 99 :89–112, 1997.

[35] Kroese D.P Mannor S. et R.Y.Rubinstein De Boer, P-T. A tutorial on the cross entropy method. *Annals of Operations Research*, 134(1) :19–67, 2005.

[36] D.P.Kroese R.Y.Rubinstein. *The Cross-Entropy Method :A Unified Approach to Combinatorial Optimisation, Monte- Carlo Simulation, and Machine Learning.* Springer, 2004.

[37] Tito Homem de-Mello et Reuven Y. Rubinstein. Rare event estimation for static models via cross-entropy and importance sampling, 2002.

[38] Raviv.T-R.Y.Rubinstein G.Alon, Kroese.D.P. Application of the cross entropy method to the buffer allocation problem in a simulation based environment. *Annals of Operations Research*, 2004.

[39] D.P. Kroese J. Keith. Sequence alignement by rare event simulation. pages 320–327, San Diego. Winter Simulation Conference.

[40] Margolin.L. *Application of the Cross Entropy method to scheduling problems.* PhD thesis, technion Industrial Engineering, 2002.

[41] Shtub.A Cohen.I, Golany.B. Managing stochastic finite capacity multi-project systems through the cross entropy method. *Annals of Operations Research*, 2004.

[42] M.Hoeck K.w.Hansmann. Production control of a flexible manufacturing system in job shop environment. pages 341–351. International Transactions in operational Reasearch, September-November 1997.

[43] M.Widmer A.Hertz. An improved tabou search approach for solving the the job shop scheduling problem with tooling constraints. *Discret Applied Mathematics*, pages 319–345, 1996.

[44] P.Michelon T.Mautor. Minausa : une nouvelle méthode combinant résolution exacte et recherche locale. page 24. Premier congrès de la société française de Recherche opérationnelle et aide à la décision (ROADF'89), Janvier 1998.

[45] E.Talbi P.Preux. Assessing the evolutionary algorithm paradigm to solve hard problems. Constraint Processing, workshop on really hard problem solving, September 1995.

[46] Bruce Schneier. *Cryptographie Appliquée*. International Thomson Publishing France, Paris, 1995.

[47] A.Sinkov. Elementary cryptanalysis. 1966.

[48] M.E.Hellman W.Diffie. Multiuser cryptographic techniques. pages 109–112. Proceedings of AFIPS National Computer Conference, 1976.

[49] M.E.Hellman W.Diffie. New directions in cryptography. *IEEE Transactions on information Theory*, IT-22(6) :644–654, November 1976.

[50] A.Shamir et L.Adleman R.Rivest. A method of obtaining digital signatures and public-key cryptosystems. *Communication of the ACM*, Février.

[51] Merkle. R. C et Hellman.M. E. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory IT-24*, pages 525–530.

[52] Brickell.E, editor. *Solving Low Density Knapsacks*. Advances in Cryptolog3, CRYPTO'88, Plenum Press.

[53] A.Shamir. A polynomial -time algorithm for breaking the basic merkle -hellman cryptosystem. *IEEE Iransactions on Information Theory. IT30*, pages 699–704, 1984.

[54] Richard Spillman. Cryptanalysts of knapsack ciphers using genetic algorithms. *CRYPTOLOGIA*, (4) :367–377, 1993.

[55] S.Micali et A.Wigderson O.Goldreich, editor. *Proofs that Yield Nothing but Their Validity ans a Methodology of Cryptographic Protocol Design*. 27th IEEE Symposium on Foundations of Computer Science, 1986.

[56] S.Micali et C.Rackoff O.Goldreich. The konwledge complexity of interactive proofs systems. *SIAM Journal on Computing*, 18(1) :186–208, 1989.

[57] L.Guillou J.Quisquater. Haw to explain zero-knowledge protocols to your children. volume 435, pages 433–444. Advances in Cryptology-Crypto'89 Springer-Verlag, 1992.

[58] Pascal véron Pierre Barthélemy, Robert Rolland. *Cryptographie : Principes et mises en oeuvres*. Edition Lavoisier France, 2000.

[59] A.Shamir. An efficient identification scheme based on permuted kernels. *Springer Verlag*, pages 606–609, 1998.

[60] A.Shamir U.Feige, A.Fiat. Zero knowledge proofs of identity. pages 210–217, Boston, Mai 1987. 19th Annual ACM Sympoosium on Theory of Computing, SSTOC'91, ACM Press.

[61] Nelson.B et Kepner.M Spillman.R, Janssen.M. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *CRYPTOLOGIA*, 17(1) :31–44, 1993.

[62] Matthews.J Robert.A. The use of genetic algorithms in cryptanalysis. *CRYPTOLOGIA*, 17(2) :187–201, 1993.

[63] Bagnall.A. *The Application of Genetic Algorithm in Crypanalysis.* PhD thesis, School of Information Systems, University of East Anglia, 1996.

[64] Andrew John Clark. *Optimisation Heuristics for cryptology.* PhD thesis, Queensland University of technology, 1998.

[65] Wikipédia. substitution box. Internet address : http ://www.answers.com/main/, 2006.

[66] Dillon J.F. *Elementary hadamard difference sets.* PhD thesis, University of Maryland, 1974.

[67] Rothaus O.S. On bent functions. *Journal of Combinatorial Theory*, A(20) :300–305, 1976.

[68] Carlet C. Two new classes of bent functions. pages 133–139. Lecture Notes in Computer Science,EUROCRYPT'94.

[69] Leander G. Dobbertin H. Crypographer's toolkit for construction of 8-bit bent functions.

[70] Fast Software Encryption (Workshop on Cryptographic Algorithms, Leuven 1994). *Construction of bent functions and balanced Boolean functions with high nonlinearity)*, volume 1008. Lecture Notes in Computer Science, Springer-Verlag.

[71] Gary McGuire Gregor Leander. Construction of bent functions from near-bent functions. *Journal of Combinatorial Theory Series A*, 116 :960–970, May 2009.

[72] Xiang-Dong Hou and Philippe Langevin. Results on bent functions. *Journal of Combinatorial Theory (A)*, 80(2) :232–246, 1997.

[73] Fatih Sulak. *Constructions of bent functions.* PhD thesis, School of applied mathematics of the middle east technical university, 2006.

[74] Timo Neumann. *Bent functions.* PhD thesis, University of Kaiserslautern, 2006.

[75] Wikipédia. Boolean function. Internet address : http ://www.answers.com/main/, 2006.

[76] Richard Hamming. error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2) :147–160, 1950.

[77] O.Staffelbach W.Meier. Nonlinearity criteria for cryptographic functions. *Eurocrypt'89*, pages 549–562, 1989.

[78] Sabine Leveiller. *Quelques algorithmes de cryptanalyse du registre filtré.* PhD thesis, l'Ecole Nationale Supérieure des Télécommunications, 2004.

[79] Ikeda Katsuo Hirose Shouichi. Nonlinearity criteria of boolean functions. 1994.

[80] Philippe Langevin, Pascal Veron, and Jean-Pierre Zanotti. Fonction booléennes équilibrées (ii). Technical report, GRIM-SCSSI, 1998.

[81] L. Welch J. Olsen, R. Scholtz. Bent-function sequences. *IEEE Transactions on Information Theory IT-28*, 6 :858–864, November 1982.

[82] J. Seberry ; X. Zhang. Highly nonlinear 0-1 balanced boolean functions satisfying strict avalanche criterion. pages 143–155. AUSCRYPT '92, December 1992.

[83] McFarland R. A family of noncyclic difference sets. *Journal of Combinatorial Theory.*

[84] Anna Grocholewska-Czurylo. A study of differences between bent functions constructed using rothaus method and randomly generated bent functions. *Journal of telecommunications and information technology*, 3 :19–24, April.

[85] P. Charpin C. Fontaine A. Canteaut, C. Carlet. Propagation characteristic and correlation-immunity of hight nonlinenar boolean function. Number 1807. Advances in Cryptographie,EUROCRYPT 2000, Springer Verlag.

[86] Susan Stepney John Andrew Clark, Jeremy L.Jacob. The design of S-Boxes By Simulated Annealing. pages 1533–1537, Portland OR, USA, June 2004. IEEE International Conference on Evolutionary Computation :CEC 2004.

[87] Subhamoy Maitra. Highly nonlinear balanced boolean functions with very good autocorrelation property. 2001.

[88] W. Stallings. *Cryptography and Network Security.* 3rd ed,Prentice Hall, 2003.

[89] Y. Zheng J. Seberry, X.M. Zhang. nonlinearly balanced boolean functions and their propagation characteristics. Advances in Cryptographie - EUROCRYPT 2000 In Advances in Cryptology - CRYPT0'93,Springer Verlag, 1994.

[90] Melek D.Yucel Selçuk Kavut. Improved Cost Function in The Design of Boolean Functions Satisfying Multiple Criteria. 2004.

[91] Ed Dawson W.Millan, Andrew Clark. Boolean Function design Using Hill climbing. *Information Reasearch Center,Queensland University of technology*, 1997.

[92] Pham Dinh Tao Pascal Bouvry Le Hoai Minh, Le Thi Hoai An. A deterministic Optimization approach for generating highly nonlinear balanced boolean functions in Cryptography. pages 381 – 392. Modeling, Simulation and Optimization of Complex Processes, Springer, 2008.

[93] Pham Dinh Tao Bouvry Pascal Le Hoai Minh, Le Thi Hoai An. A Combined DCA - GA for Constructing Highly Nonlinear Balanced Boolean Functions in Cryptography. *Journal of Global Optimization, Online first 10.1007/s10898-009-9481-4.*

[94] Hoai An Le Thi Sarra Bouallagui and Tao Pham Dinh. Design of highly nonlinear balanced boolean functions using an hybridation of dca and simulated annealing algorithm. volume 14, pages 583–592. Modeling, Computation and Optimization in Information Systems and Management Sciences, Communications in Computer and Information Science CCIS, Springer, 2008.

[95] L. Wolsley G. Nemhauser. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

[96] J.A. Clark and J. L. Jeremy. Two-stage Optimisation in the Design of Boolean Functions. volume 1841, pages 242 – 254. Lecture Notes In Computer Science ; Proceedings of the 5th Australasian Conference on Information Security and Privacy, 2000.

[97] Rubinstein.R Homem-de Mello.T. Rare event estimation for static models via cross-entropy and importance sampling. *Annals of Operations Research*, 2002.

[98] Pham Dinh Tao Le Thi Hoai An, Le Hoai Minh and Pascal Bouvry. Solving the perceptron problem by deterministic optimization approach based on dc programming and dca. pages 222–226. Proceeding of the 7th IEEE International Conference on Industrial Informatics, 2009.

[99] Bouallagui Sarra Le Thi Hoai An, Pham Dinh Tao. Cryptanalysis of an identification scheme based on the perceptron problem using a hybridization of deterministic optimization and genetic algorithm. pages 117–123. Proceedings of the International Conference on Security and Management, World Congress in Computer Science Computer Engineering, and Applied Computing, July 13-16 2009.

[100] Lars.R.Knudsen and Willi Meier. Cryptanalysis of an identifcation scheme based on the permuted perceptron problem. Eurocrypt'99, 1999.