Méthode de conception fonctionnelle en architecture : une approche CAO basée sur les contraintes : ARCHIPLAN

Benachir Medjdoub

THÈSE DE DOCTORAT

DE

L'ÉCOLE CENTRALE DE PARIS

Spécialité :
PRODUCTIQUE

Présentée
à l'École Centrale de Paris
par

Benachir MEDJDOUB

pour obtenir le grade de

DOCTEUR DE L'ÉCOLE CENTRALE DE PARIS

Sujet de la thèse :

"MÉTHODE DE CONCEPTION FONCTIONNELLE EN ARCHITECTURE : une approche CAO basée sur les contraintes : ARCHiPLAN"

Soutenance le 7 mai 1996

devant le jury composé de :

Paul QUINTRAND	Président	(GAMSAU/CNRS)
Jean-Marc BRUN	Rapporteur	(Université Claude Bernard, Lyon1)
Michel LÉGLISE	Rapporteur	(École d'architecture de Toulouse)
Bernard YANNOU	Examinateur	(École Centrale Paris)
Robert MACULET	Examinateur	(Université de Provence)
Jean-Claude BOCQUET	Examinateur	(École Centrale Paris)
Ian SMITH	Examinateur	(École Polytechnique Fédérale de Lausanne)

Laboratoire Productique-Logistique
École Centrale de Paris
Grande Voie des Vignes
92295 CHATENAY-MALABRY (FRANCE)

A mes parents,
A toute ma famille,

Remerciements

Je tiens à remercier avant tout M. Jean-Claude Bocquet, professeur à l'École Centrale de Paris, de m'avoir accueilli au laboratoire Productique et Logistique et de m'avoir fourni un cadre intellectuel et matériel durant ces années de thèse.

Je tiens à exprimer ma profonde gratitude à M. Bernard Yannou, pour l'encadrement de cette thèse, pour ses conseils et ses encouragements.

Toute ma reconnaissance va à M. Michel Léglise et M. Jean-Marc Brun pour avoir accepté le rôle délicat de rapporteur. Je tiens à remercier M. Paul Quintrand et M. Ian Smith d'avoir accepté d'être membres du jury. Je tiens également à remercier M. Robert Maculet pour le soutient qu'il m'avait apporté au laboratoire STB et d'avoir accepté d'être membres du jury.

Je tiens également à remercier M. Philippe Charman et M. Bertran Neuveu d'avoir gracieusement mis à notre disposition le logiciel EAAS.

Je voudrais également faire part de ma très grande reconnaissance envers, Mme. Annie Chahin, Mme. Anne Prévot, M. Patrick Callet, M. Daniel Letteron, M. Jean-Loup Burgaud et M. Boussad Mammeri.

Je tiens également à remercier toute l'équipe du laboratoire Productique Logistique pour la sympathie qu'elle m'a témoignée et les encouragements qu'elle m'a prodiguée.

Benachir Medjdoub

Table des matières

7 Conclusion

Annexes

Chapitre 1

Introduction

1.1. Motivations

Sur un projet architectural, le premier geste de l'architecte consiste à établir un cahier des charges[1] à l'issue d'une consultation du maître d'ouvrage et/ou des clients. Par la suite, les architectes abordent traditionnellement les premières phases de la conception (conception préliminaire) en crayonnant plusieurs esquisses à « main levée » en accord avec le cahier des charges et qui traduisent des schémas de principe de placement[2]. Ce n'est qu'une fois une esquisse établie que la CAO prend le relais pour servir de support et d'aide au dimensionnement (conception détaillée), au chiffrage des devis, à la gestion des projets, etc.

Il est déroutant de constater qu'à l'heure actuelle, les architectes résolvent ces problèmes de placement « à la main ». En effet, ils se servent, en général, des logiciels de CAO architecture comme de « planches à dessin électroniques ». À ce jour il n'y a aucune approche générale Assistée par Ordinateur de conception fonctionnelle, c'est-à-dire de prise en compte du cahier des charges ni d'intégration de la conception préliminaire au sein d'un logiciel commercialisé.

La motivation de cette thèse était donc de développer une méthode de conception fonctionnelle en architecture (logement, équipements collectifs, etc) qui tienne compte des spécifications fonctionnelles d'un bâtiment dès les premières phases de conception. Cette méthode et l'outil qui en découleraient ne devaient en aucun cas se substituer à l'architecte, mais devaient l'assister dès les premières phases de conception en vue d'optimiser la conception au sens de la qualité et des coûts.

1.2. Objectifs de cette recherche

Dans l'idéal, une approche fonctionnelle pour un problème de placement doit permettre au sein d'un logiciel de CAO, de modéliser les contraintes fonctionnelles sur les objets architecturaux, puis doit mettre en œuvre des algorithmes qui aident au maximum le concepteur à visualiser les « champs du possible », c'est-à-dire l'espace des solutions de placement potentielles. Enfin, un tel logiciel, loin de donner une solution unique, doit permettre à un concepteur de « naviguer » dans cet espace du possible, il doit lui laisser la possibilité d'étudier de manière précise plusieurs alternatives de solution, de classer les solutions par ordre préférentiel en justifiant leur qualité, d'affecter des priorités et des degrés d'incertitude aux contraintes[3], de

1. Les architectes utilisent préférentiellement le terme de *programme*.
2. On dit aussi *composition* ou *allocation spatiale*.
3. On peut ainsi aborder le problème de réhabilitation au moindre coût en tentant de réutiliser le maximum de cloisons existantes.

tester de manière différentielle la relaxation de certaines contraintes. Donc, loin d'être un logiciel « boîte noire », hermétique, un logiciel de CAO fonctionnelle en architecture doit permettre, tout en bornant à tout moment l'espace potentiel des solutions, de répondre aux questions de haut niveau de sémantique que se pose le concepteur.

La recherche en allocation spatiale a une histoire assez ancienne qui remonte aux débuts de l'informatique : les années 70. De nombreux travaux se heurtent à l'explosion combinatoire des solutions de placement et plusieurs d'entre eux conclurent d'emblée à l'utopie de ce domaine de recherche. De ce fait, la recherche stagne pendant une dizaine d'années. Elle redémarre avec les approches systèmes experts puis, plus récemment, avec la programmation par contraintes. Pourtant, toutes ces approches n'échappent pas à deux problèmes en apparence insurmontables : **l'explosion combinatoire des solutions** et **la pauvreté des contraintes**.

En effet, aucune des approches antérieures [Eastman, 1973 ; Pfefferkorn, 1975 ; Flemming, 1988 ; Maculet, 1991 ; Charman, 1995] n'a vraiment résolu le problème récurrent de l'explosion combinatoire des solutions numériques de placement. En effet, lors d'une *énumération* de toutes les solutions de placement ou lors d'une recherche de la meilleure solution (*optimisation*), deux solutions quasi-identiques où seule une cloison a été translatée d'un *module*[4] sont considérés comme deux *solutions numériques* différentes. Il est clair qu'en phase de conception préliminaire il est, d'une part, inutile de discriminer deux solutions numériques proches et, d'autre part, cela provoque une explosion de solutions[5]. Cette première raison est suffisante pour expliquer les impossibilités d'utilisation concrète de ces premières approches.

La seconde raison pour laquelle les tentatives d'approche fonctionnelle ont échoué est le manque de souplesse dans l'expression des contraintes ainsi que la non-exhaustivité des contraintes fonctionnelles de placement qui existent. Faire un outil de conception général et ouvert est une difficulté de taille que tous les outils de conception devraient surmonter. De nombreuses tentatives de conception fonctionnelle en architecture se sont fondées sur des systèmes experts [André, 1986 ; Flemming,1988]. Cette approche présente de nombreux inconvénients : nous ne sommes jamais sûr de l'exhaustivité et de la cohérence des règles, nous ne sommes jamais sûr d'obtenir l'optimum global et les temps de réponse sont importants.

---

4. Un architecte appelle module l'incrément de distance dont sont multiples toutes les dimensions de locaux (largeur, longueur).
5. typiquement plusieurs milliers, voire plusieurs millions.

Pour pallier les limites des méthodes utilisées, nous avons tout d'abord sur le plan méthodologique :

- (1) **Proposé un modèle de conception à trois niveaux** dont un niveau intermédiaire de solutions dites *topologiques*, entre les spécifications fonctionnelles et les solutions numériques (cf. Figure 1.1). Une solution topologique est une classe d'équivalence de solutions numériques respectant les mêmes conditions d'orientation relative entre tous les couples de locaux. Ainsi, deux solutions *topologiquement différentes*, se différencient par au moins une orientation relative différente. Nous avons montré que la représentation graphique d'une solution topologique correspond à une *esquisse*, c'est-à-dire aux dessins à main levée que l'architecte effectue en phase de conception préliminaire, sans dimension numérique précise, mais qui laisse espérer toutefois qu'il existe au moins une solution numérique respectant les contraintes. L'avantage de ce niveau de solutions topologiques est le faible nombre de solutions existantes[6].

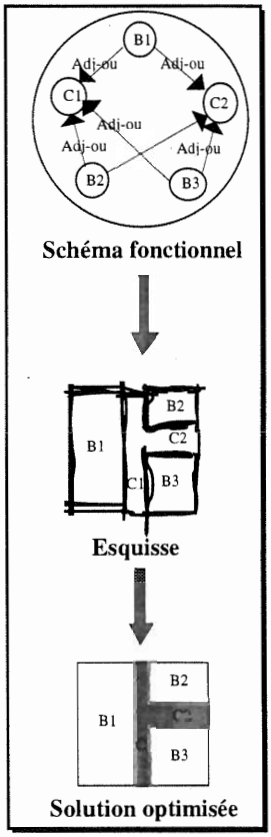


Figure 1.1 : Les trois niveaux de conception proposés dans notre approche

La principale valeur ajoutée de notre travail de recherche est ce modèle à trois niveaux qui a permis à la fois de : résoudre pour une bonne part (les tests le

6. de l'ordre de quelques dizaines.

montrent) le problème de l'explosion combinatoire et de s'adapter au mode de travail des architectes en conception préliminaire (phase d'esquisse). Ce contournement de l'explosion combinatoire s'explique par le fait qu'on découpe un gros problème contraint d'énumération de toutes les solutions de placement en une première énumération de solutions topologiques (et donc en sous-problèmes contraints) et en la recherche de la meilleure solution de placement pour les solutions topologiques choisies.

- (2) **Optimisé les solutions numériques** afin d'obtenir les meilleures solutions au regard de critères d'architecture (surface de circulation, longueur de murs, etc) à minimiser.
- (3) **Généralisé notre méthode à des bâtiments à plusieurs étages**, grâce au modèle de connaissances architecturales, et des algorithmes de recherche de solution que nous avons proposés.
- (4) **Proposé des solutions pour l'interface Homme/machine.**

Sur le plan technique :

- (1) Il a été montré depuis plusieurs années que les techniques de **Programmation par contraintes** (PPC) apportaient, d'une part, une grande souplesse dans l'utilisation des contraintes puisque la définition des contraintes est découplée des algorithmes de résolution, d'autre part, ces techniques permettent d'aborder des problèmes de conception hautement combinatoires comme c'est le cas pour le placement optimal [Maculet, 1991 ; Aggoun, 1992 ; Flemming et all, 1992 ; Charman, 1995]. Nous montrerons que les techniques de Programmation par contraintes réalisent une alternative des plus prometteuses aux techniques paramétriques et variationnelles des modeleurs de CAO.
- (2) les techniques de programmation par contraintes combinées à la **programmation orientée objets** nous ont permis de définir des classes hiérarchiques de locaux incluant des contraintes spécifiques[7].

1.3. Organisation

Dans le *chapitre 2*, nous présentons, après une analyse du processus de conception en architecture et un état de l'art en CAO, les spécifications de notre travail de thèse. Le *chapitre 3* porte sur le modèle architectural composé des objets architecturaux et des contraintes de placement. Les algorithmes d'énumération des solutions topologiques

7. On parlera de *contraintes de classes*.

et d'optimisation des solutions numériques seront présentés au *chapitre 4*. L'aspect interface Homme/Machine sera présenté au *chapitre 5*. Enfin, plusieurs études de cas (logement, maison individuelle, jardin d'enfant, etc) valident notre approche au *chapitre 6*.

1.4. Mise en œuvre

Notre approche de conception fonctionnelle a été mise en œuvre au sein du logiciel ARCHiPLAN. ARCHiPLAN est développé au dessus de la plate-forme d'intelligence artificielle de la société ILOG. Cette plate-forme est basée sur le langage Le-Lisp[8] qui est un langage Lisp orienté objet. Cet environnement intègre des fonctionnalités graphiques (AÏDA[9]) parmi lesquelles la bibliothèque GRAPHER de manipulation de structures de graphes. Il intègre également la bibliothèque de Programmation par contraintes PECOS.

ARCHiPLAN a été développé et testé sur une station de travail IBM RS6000 320H (30 Mips, 8,5 Mflops), les temps d'exécution des algorithmes donnés dans cette thèse s'y rapportent donc. De plus, il est à noter que l'exécution du code Le-Lisp s'est fait en mode interprété et non compilé, la compilation pouvant apporter un gain de temps d'un facteur 4 à 10.

8. Le-Lisp est une marque déposée de l'INRIA.
9. AÏDA et PECOS sont des marques déposées de la société ILOG [Ilog, 1991 ; Puget et al, 1991].

Chapitre 2

L'architecture et la CAO

« L'architecture est un fait d'art, un phénomène d'émotion, en dehors des questions de construction, au delà. La construction, c'est pour faire tenir ; l'architecture, c'est pour émouvoir. »

Le Corbusier, Vers une architecture 1923.

2.1. La problématique

La CAO fait désormais partie de l'environnement courant des architectes et des concepteurs en général. Le progrès technique et la créativité des concepteurs de logiciels, interviennent dans cette dynamique. Ils l'alimentent en aides au travail, prenant tour à tour le parti de l'instrumenter en totalité, ou tâche par tâche, ou bien encore en essayant d'automatiser certains de ses raisonnements opérationnels. En architecture l'informatique est utilisée essentiellement au niveau de la représentation (images de synthèse), de la production du projet architectural (production des différents plans, etc), ou encore dans les phases techniques (calcul de structures, ventilation, thermique, gestion du projet, etc). Actuellement, aucun système existant n'assiste le concepteur dès les premières phases de conception ou dès la spécification du problème. L'élaboration de ce type de système ne pourra se faire qu'en intégrant le processus de conception préliminaire en architecture. ARCHiPLAN a été conçu dans cet objectif.

2.1.1. Les différentes acceptions de l'architecture

À l'origine, le concept d'*architecture* est attaché aux objets bâtis, il couvre en fait aujourd'hui un domaine d'objets beaucoup plus large (on parle également de l'architecture des ordinateurs, etc). *Hanrot* [Hanrot, 1989], dans sa thèse, a défini les trois acceptions de l'architecture que nous adopterons :

• **(1) Architecture comme objet** : « *l'architecture est un ensemble d'objets architecturaux (les édifices, les bâtiments, les fabriques, les abris et leurs parties) qualifiés par des propriétés fonctionnelles, morphologiques, constructives, techniques, économiques, esthétiques, symboliques, établis dans un site, et ordonnancés du tout aux parties et des parties au tout* ».

• **(2) Architecture comme art du projet** : « *l'architecture est l'art de la description d'une architecture objet qui n'existe pas dans le monde réel, en vue de sa réalisation, de son usage et de sa cohérence contextuelle (site, culture, technique et économie, etc)* ».

Cette définition rejoint celle de *Pérouse de Montclos* [Pérouse, 1972] qui est : « *Architecture, art de construire les édifices et d'aménager les jardins* ».

• **(3) L'architecture comme savoir** : « *le savoir architectural est la somme des connaissances élaborées sur l'architecture et sur le projet* ».

Viollet-le-Duc a donné de l'architecture la définition suivante : « *L'architecture se compose de deux éléments, la théorie et la pratique* ». La pratique renvoie à la définition de l'architecture comme art du projet, et la théorie comme une formalisation du savoir en

architecture. La théorie est entendue comme une forme de connaissance abstraite, comme représentation de la pratique. Les textes *théoriques* d'architecture prennent leur distance par rapport à la pratique et prétendent à une certaine universalité. Ils se donnent comme but de mettre en forme le savoir architectural, c'est-à-dire qu'ils proposent un énoncé organisé et succinct qui permet de rendre compte de sa complexité avec économie. Dans le passé, de nombreux architectes et penseurs ont essayé de théoriser le savoir architectural, parmi les travaux théoriques qui ont le plus marqué l'histoire de l'architecture, citons :

- Ceux de *Vitruve* l'architecte romain, décédé en l'an 29 av. JC. Ayant servi César en Gaule et en Espagne, il a écrit le premier ouvrage sur l'architecture dont on ait conservé des traces : « *L'architecture est un art et une science* ». *Vitruve* distingue différents niveaux de composition en architecture et a surtout développé la théorie des trois ordres de l'architecture classique : « *Dorique, Ionique et Corinthien* ». Des règles précises accompagnent ces ordres pour l'établissement d'une façade ou d'un plan.

- ceux de *Durant*, architecte et archéologue, professeur à l'École polytechnique au 19ème siècle a formé plusieurs générations d'élèves à sa méthode de conception en architecture [Durant, 1801].

- ceux de *Viollet-Le-Duc* (1814-1879) qui a développé sa théorie dans ses « *Entretiens sur l'architecture* ».

- ceux d'*Arnaud*, [Arnaud, 1928] architecte et professeur à l'École centrale de Paris, dans son cours de 1925 intitulé « *La composition en architecture* » a décrit le processus de conception préliminaire d'un bâtiment en prenant un exemple concret concernant le déplacement de l'École centrale de Paris.

- ceux de *Lurçat*, architecte-urbaniste français du mouvement fonctionnaliste qui entre 1937 et 1944 a écrit un ouvrage de référence sur la théorie en architecture : « *Formes, composition et lois d'harmonie* » [Lurçat, 1955].

- bien entendu, les travaux de *Le Corbusier* qui, pour l'architecture moderne, reste une légende bien inscrite dans notre siècle. Les travaux de *Le Corbusier* sont innombrables, nous nous contenterons de citer la définition du *modulor*, système de mesure basé sur le corps humain que *Le Corbusier* a souhaité appliquer à l'architecture et à la mécanique : « *Le Modulor est un langage des proportions qui rend compliqué le mal et simple le bien* » [CGP, 1987],

  Plutôt que le *Modulor*, nous utilisons dans ARCHiPLAN la notion de *module* telle que définie par *Viollet-le-Duc* comme une « *unité de convention, mesure arbitraire servant à établir les proportions des parties d'un édifice* ».

- ceux de *Neufert* [Neufert, 1967 et 1983], architecte et professeur à l'École Supérieure Technique de Darmstadt a établi, dans un souci de réglementer et de standardiser la construction, une encyclopédie très utilisée par les élèves et les architectes lors de la conception d'un projet architectural.
- l'une des dernières tentatives de l'approche scientifique de la conception en architecture fut celle *d'Alexander*, architecte et enseignant à l'université de Berkeley dans son ouvrage « *La synthèse de la forme* » [Alexander, 1971 ; Élalouf, 1974], *Alexander* met en relation l'adaptation réciproque de la forme en architecture et du contexte dans lequel elle est bâtie, c'est-à-dire l'adaptation de la forme au contexte et du contexte à la forme, contrairement à d'autres objets manufacturés comme une automobile qui est conçue indépendamment de son environnement.
- les travaux de Boudon [Boudon, 1971] sur une architecturologie qui serait une science de l'artificiel, ayant l'architecture comme objet d'étude.

Les quelques théories que nous avons citées montrent bien qu'il n'existe pas de théorie unifiée en architecture contrairement aux sciences exactes ou humaines.

2.1.1.1. Les objets architecturaux

Les objets architecturaux renvoient à la définition de l'architecture comme objet. Ce sont eux qui sont manipulés durant la conception d'un projet et constituent le support de l'architecture. La modélisation des connaissances architecturales passe par la modélisation des objets architecturaux. Les objets architecturaux peuvent être classés en différents niveaux, *Maculet* [Maculet, 1991] a proposé la classification suivante :

- Niveau 1 : L'ensemble de bâtiments (grand-ensemble, ensemble-historique, etc),
- Niveau 2 : Le bâtiment : entité, corps de bâtiment, bâtiment ou groupe de bâtiments (immeuble d'habitation, hôpital, usine, etc),
- Niveau 3 : Les espaces architecturaux : division d'un bâtiment (escalier, étage, logement, pièce, etc),
- Niveau 4 : Les éléments d'architecture : les éléments de formes pleines, les éléments constructifs (murs, cloison, baie, porte, fenêtre, etc),
- Niveau 5 : Les constituants : unité de matériaux de construction (brique, parpaing, etc).

Dans ARCHiPLAN nous nous situons au niveaux 3, nous parlerons essentiellement d'espaces architecturaux. Nous reviendrons en détail sur les espaces architecturaux au *chapitre 3*.

2.1.1.2. Les relations spatiales en architecture

La conception en architecture est en partie de fixer des relations spatiales entres les objets architecturaux. Nous retrouvons différents concepts dont celui de *contiguïté*, d'*accolement*, d'*adjacence*, de *mitoyenneté* où encore de *calage*. Ces concepts sont liés au niveau de définition des objets architecturaux mis en relation. Le*calage* concerne les relations spatiales entre les éléments d'architecture (poteaux, poutres, etc), il a été largement développé au GAMSAU[10] [Quintrand et al, 1985]. Dans ARCHiPLAN, les relations spatiales concernent les relations entre les espaces d'architecture qui se retrouvent essentiellement dans les concepts d'*adjacence*, d'*inclusion* et de *recouvrement*. Nous considérons ces relations spatiales comme des *contraintes spatiales* qui permettent de décrire le cahier des charges d'un bâtiment. Nous décrirons en détail le modèle de *contraintes spatiales* d'ARCHiPLAN dans la deuxième partie du *chapitre 3*.

2.1.1.3. Le modèle de composition

Le modèle de composition regroupe les éléments qui permettent de réguler la disposition des objets architecturaux. Parmi les principaux éléments de régulation nous comptons les trames et les axes.

• Les trames : Selon *Quintrand* [Quintrand et al, 1985], « *la trame est un système d'axes et fonctionne comme telle, c'est-à-dire comme un système référentiel, mais elle désigne un système d'axes dont la distribution est systématique et qui vise généralement à discrétiser l'espace à des fins multiples, unification, standardisation, coordination dimensionnelle. La trame constituera un puissant outil de coordination des calages des éléments (en particulier avec l'usage des conventions de coordination dimensionnelles)* ». Trois catégories de trames ont été définies :

- la trame de localisation (espace de localisation discrétisé). C'est une grille projetée sur un plan.
- la trame de structures détermine la localisation et le calage d'éléments porteurs.

10. GAMSAU : Groupe d'études pour l'Application des Méthodes Scientifiques à l'Architecture et l'Urbanisme.

- les trames fonctionnelles permettent la gestion des localisations de sous-ensembles fonctionnels (espaces) ou éléments fonctionnels.

Nous avons illustré dans les Figure 2.1 et Figure 2.2 la perspective et le plan de masse d'une maison [Neufert, 1967], ce projet est basé sur un tracé régulateur (axes et trames).

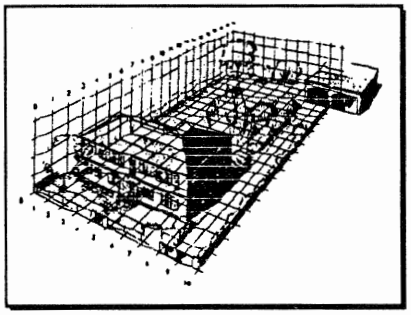


Figure 2.1 : Perspective d'une maison d'habitation [Neufert, 1967]

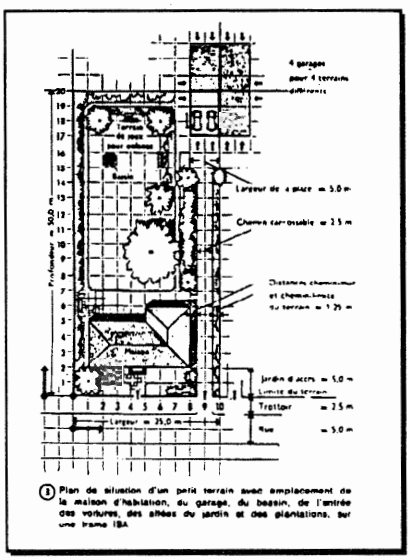


Figure 2.2 : Plan de masse de la maison d'habitation (Figure 2.2) [Neufert, 1967]

- Les axes : *Maculet* [Maculet, 1991] distingue deux types d'axe :
  - Les axes d'ensemble et de division qui partagent et ordonnent les ensembles de la division des bâtiments ; ce sont les lignes principales du plan,
  - Les axes des éléments qui sont des axes de symétries locales.

Lurçat a [Lurçat, 1955] a décrit le principe d'organisation des objets architecturaux :

- la répétition,
- le rythme,
- l'échelle,
- le contraste.

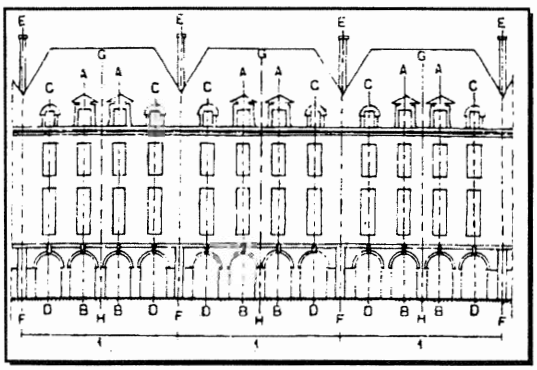


Figure 2.3 : Ordonnance de rythmes de la place des Vosges (Paris), extrait de [Lurçat, 1955]
— axe de symétrie de chaque bâtiment : *G-H*,
— trame de distribution des travées déterminant un jeu d'axes locaux : *C-D* et *A-B*,
— trame de distribution des bâtiments : *E-F*.

Dans ARCHiPLAN, le modèle de composition se limite à la prise en compte des trames que nous retrouverons au *chapitre 4*.

2.1.2. Le processus de conception

Le processus de conception couvre l'ensemble des étapes de la production d'un bâtiment allant du premier contact entre l'architecte et le client jusqu'à l'établissement des *plans d'exécution*[11]. L'analyse du processus de conception est une démarche nécessaire pour pouvoir comprendre les interactions entre les différentes étapes et la logique d'élaboration du projet architectural. Notre objectif étant de prendre en charge ces premières étapes de conception par l'outil informatique, il nous était indispensable non seulement de comprendre mais de formaliser les étapes de la conception d'un bâtiment.

2.1.2.1. Quelques travaux du domaine

Le processus de conception architecturale est un problème complexe à cause de l'imbrication de plusieurs domaines : techniques, économiques, artistiques, etc.

Quintrand [Quintrand et al, 1985] a défini le processus de conception en trois étapes :

- **(1) l'intention** : c'est la phase de préparation de l'information (physiques, sociales, juridiques, économiques, etc) nécessaire à la réalisation du projet. Ces informations sont regroupées dans ce qui est communément appelé le *programme* ou le *cahier des charges*. Cette phase se solde par des *organigrammes* ou *schémas fonctionnels*. Un *schéma fonctionnel* représente les relations topologiques entre les espaces, c'est-à-dire les relations de voisinage et de

11. Les plans d'exécution et les plans techniques sont ceux qui servent à la réalisation des ouvrages [DICOBAT, 1991].

communication, les espaces sont représentés sous forme de cercles et les relations par des traits.

- **(2) la projettation**[12] : c'est la phase où le projet d'architecture est développé. Elle est divisée en deux sous-étapes :
  - **(2-1) la conception** : c'est la phase principale de la création architecturale. Cette sous-étape est caractérisée par la composition architecturale[13] et l'élaborations des esquisses.
  - **(2-2) l'instrumentation**[14] : qui consiste à rendre faisable le projet, l'architecte met à l'échelle[15] le projet en tenant compte des aspects techniques. On appelle aussi cette étape *mise à l'échelle.*
- **(3) la réalisation** : cette phase concerne l'architecte dans ses missions de coordination et de contrôle du chantier.

Lebahar [Lebahar, 1983], quant à lui, a défini le processus de conception à travers le dessin comme outil de simulation des problèmes et des solutions, il l'a divisé en trois étapes :

- **(1) le diagnostic architectural** : c'est la première étape de conception, elle se solde par quelques notes écrites et un premier dessin ou groupe de dessins appelé par *Lebahar "base graphique de simulation".*
- **(2) la recherche de l'objet par simulation graphique** : cette phase consiste en l'enrichissement de la *base graphique de simulation*. C'est pendant cette phase que l'architecte résout le problème de placement des espaces.
- **(3) l'établissement du modèle de construction** : c'est la coordination dimensionnelle du projet, elle consiste à définir avec précision l'ensemble des représentations graphiques du projet.

Nous avons illustré dans la Figure 2.4 ce que *Lebahar* a appelé *le schéma général de la recherche de l'objet par simulation graphique* où il est indiqué les états intermédiaires lors de l'élaboration du projet architectural. *Lebahar* a représenté deux dimensions. L'une est *synchronique*, elle concerne les éléments stables dont on tient compte à chaque état de la résolution d'un problème architectural et l'autre est *diachronique*, elle concerne la

12. Néologisme de *Quintrand* [Quintrand et al, 1985].
13. « *Composer, c'est grouper des éléments choisis pour en faire un tout homogène et complet, de telle sorte qu'aucune partie de ce tout ne puisse prétendre se suffire à elle-même, mais que toutes, au contraire, se subordonnent plus ou moins à un élément commun d'intérêt, centre et raison d'être de la composition. L'architecture est l'expression concrète d'une idée. Cette idée doit être franche, unique et clairement exprimée* » (Gromort) [Duplay, 1985].
14. Néologisme de *Quintrand* [Quintrand at al, 1985].
15. L'échelle est le rapport entre deux systèmes dimensionnels, par exemple le système de construction d'un bâtiment et son système de représentation.

progression de ces éléments dans le temps et de certains rapports qu'ils entretiennent entre eux. Dans la Figure 2.5, (PA) représente le problème architectural, (A) est le système de traitement de l'information architecturale, (S) est le modèle de simulation, et (O) est le modèle d'objet. Chaque état intermédiaire est défini par *Lebahar* comme suit : l'architecte transforme l'état de l'objet contenu dans (PA) en un autre état (O) après avoir testé, modifié, et précisé ses hypothèses de solutions dans une représentation graphique (S) spécialement établie pour cela. Dans notre approche, ARCHiPLAN procède d'une manière analogue à celle citée par *Lebahar*. Au fur et à mesure de la résolution du problème, les éléments et les relations entre eux sont précisés jusqu'aux solutions finales à travers ce qui sera nos trois niveaux de conception.

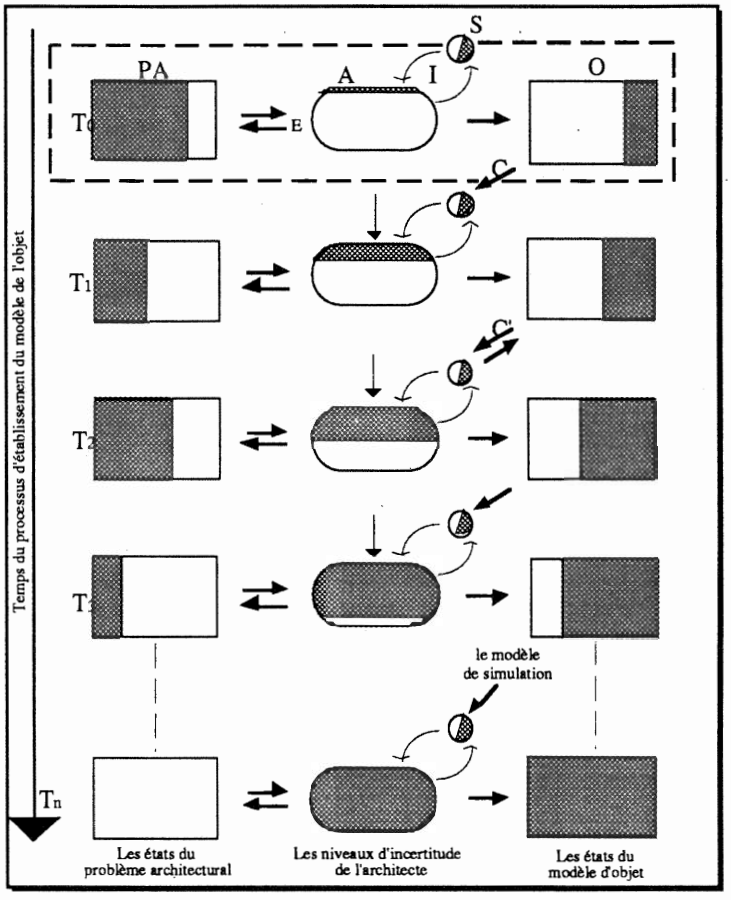


Figure 2.4 : Schéma général de la recherche de l'objet par simulation graphique[Lebahar, 1983]. Le grisé représente la précision de l'objet au cour du processus.

Les travaux de *Quintrand* et de *Lebahar* sont complémentaires dans le sens où *Lebahar*, met en œuvre le processus de conception à traver le dessin que fait l'architecte.

Dans [Coyne et Gero, 1990], le processus de conception est représenté en trois niveaux d'abstraction illustrés dans la Figure 2.5. Ces trois niveaux représentent un *schéma*

fonctionnel, une *esquisse* et un plan mis à l'échelle. Nous retrouvons ces trois niveaux dans ARCHiPLAN.

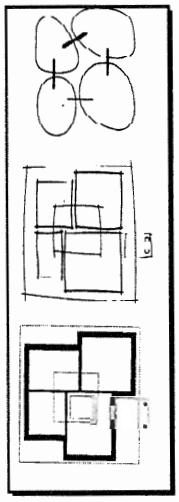


Figure 2.5 : Les trois niveaux de conception de [Coyne et Gero, 1990] : schéma fonctionnel, esquisse et plan

Dans chaque étape du processus, l'architecte manipule des informations qui se précisent au fur et à mesure qu'il avance dans le temps. Contrairement à la forme linéaire (partition en phases) représentée dans le processus, celui-ci est itératif. D'ailleurs on parle souvent « *d'aller et retour* » entre les différentes phases, ceci nous semble être une expression réductrice et simplificatrice de la véritable complexité de ce problème. Chaque fois que le concepteur revient à une phase précédente, ça se solde toujours par un état nouveau, étant donné que les raisons d'un retour en arrière enrichissent le processus de conception et l'expérience même de l'architecte. C'est une forme d'autocritique de son propre projet.

2.1.2.2. L'exemple d'un jardin d'enfants

A partir du *cahier des charges* d'un jardin d'enfants (proposé par *Maculet* [Maculet, 1991]), nous allons illustrer chaque étape de conception et voir les différents systèmes de représentation utilisés dans chacune d'elles. L'important, dans cet exemple, est la démarche et les différents systèmes de représentation utilisés.

- **Définition du cahier des charges**

Deux cas de figures peuvent se présenter. Le premier cas touche les grands projets où, en général, un cahier des charges est déjà établi et remis à l'architecte. Dans le deuxième cas, c'est à l'architecte d'établir le *cahier des charges* selon les recommandations et les informations qu'il a recueillies auprès de son client. La partie

du cahier des charges portant sur les contraintes spatiales du jardin d'enfants est donnée dans le Tableau 2.1. Les surfaces bien qu'apparemment instanciées ne sont pas considérées comme telles par un architecte car, traditionnellement, il s'autorise jusqu'à une augmentation de 5%.

Activités		Surface (en module)	Activités		Surface (en module)
1	Classe 1	32	13	Salle de réception	5
2	Classe 2	32	14	Toilette	2
3	Classe 3	32	15	Bureau des institutrices	4
4	Classe 4	32	16	Entrée	10
5	Classe 5	32	17	Salle de rangement	2
6	Patio	60	18	Salle de rythmique	50
7	Préau couvert	70	19	Vestiaire 1	4
8	Salle de repas	50	20	Vestiaire 2	4
9	Cuisine	10	21	Vestiaire 3	4
10	Salle d'eau	10	22	Vestiaire 4	4
11	Infirmerie	4	23	Vestiaire 5	4
12	Bureau de la directrice	4			
1 module de surface = 3 m^2					

Tableau 2.1 : *Cahier des charges* du jardin d'enfants

- **L'établissement du schéma fonctionnel**

Parmi les premiers actes graphiques de l'architecte, nous retrouvons les schémas et organigrammes. Ils représentent une solution d'agencement des circulations avec les différentes activités du projet : c'est ce que *Maculet* [Maculet, 1991] a appelé la *conception logique*. Cette représentation demande également une hiérarchisation des différentes *activités* qui composent le programme (surtout dans le cas d'un programme important : bibliothèque, théâtre, installation industrielle, etc). Cette hiérarchisation permet également de dégager les différents niveaux de conception. Nous pouvons citer au minimum deux niveaux :

- (1) **niveau 1** : le *plan de masse*[16] (on manipule des ensembles d'*activités*) ;
- (2) **niveau 2** : le niveau des *activités* (où on manipule les activités du programme).

Le regroupement des activités du *cahier des charges* en plusieurs ensembles et sous-ensembles permet de réduire le nombre d'éléments manipulés par le concepteur et par conséquent la complexité du problème. Ces ensembles regroupent des *activités* de même nature qui font appel à une analyse de chacune d'elles dépendant de deux facteurs :

16. Le *plan de masse* est un plan à échelle réduite (souvent au 1:200) qui situe les implantations d'un ou plusieurs bâtiments sur leurs terrain et dans leur environnement [DICOBAT, 1991].

- (1) **le programme** (taille, complexité, etc),
- (2) **le concepteur** (son savoir faire, son expérience et son *parti pris*).

Nous avons regroupé les activités en cinq sous-ensembles, un nom évocateur de la fonction de l'espace a été attribué a chaque sous-ensemble (les numéros référencent les *activités* définies dans le *cahier des charges*) :

- [A] (*Classes*) = {{1, 2, 3, 4, 5},{19, 20, 21, 22, 23}}
- [B] (*Administration*) = {11, 12, 13, 14, 15}
- [C] (*Réfectoire*) = {8, 9, 10}
- [D] (*Circulation*) = {6, 7, 16}
- [E] (*Sport/Gym*) = {17, 18}

Nous avons illustré dans la Figure 2.6 le schéma fonctionnel du jardin d'enfants pour lequel nous avons considéré deux niveaux de conception : le *plan de masse* et le niveau des *activités*. Nous remarquerons que ce dessin représentant un graphe avec des nœuds et des relations entre ces nœuds, est une expression graphique encore embryonnaire de l'idée de l'architecte. Ces relations ne traduisent pas des relations précises de positionnement relatif (orientation et distance), mais plutôt des relations moins contraignantes, d'ordre topologique (adjacence, non-adjacence, adjacence au nord, etc). Nous verrons au *chapitre 4* comment au niveau informatique nous allons intégrer cette phase très importante du processus de conception.

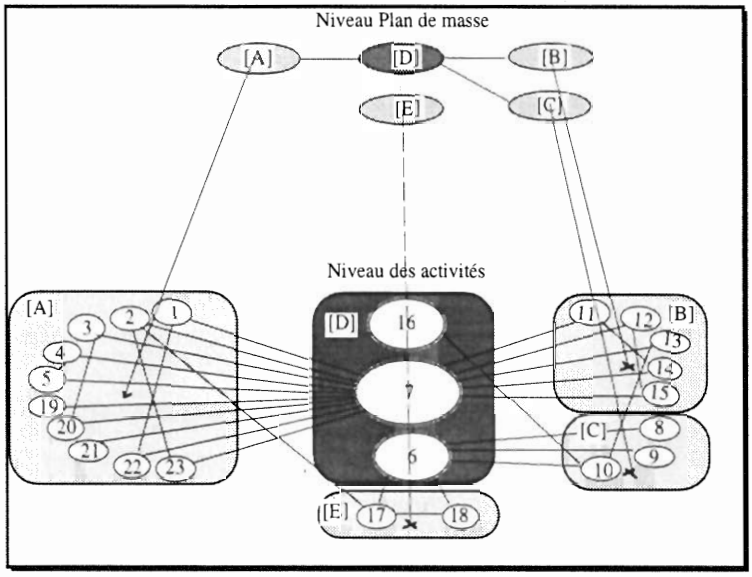


Figure 2.6 : *Schéma fonctionnel* (*plan de masse* et niveau des *activités*)

- **L'établissement des esquisses et la composition en architecture**

C'est la phase principale de la création architecturale. L'étape d'esquisse est une étape de créativité ou l'architecte propose des principes de représentation géométrique sans grande précision dimensionnelle mais en respectant toutefois le schéma fonctionnel (cf. Figure 2.7). Elle est le résultat de l'articulation entre la *modélisation logique* (schémas fonctionnels) et la modélisation géométrique dont le dessin est le principal support. C'est au cour de l'élaboration des esquisses que l'architecte procède à ce qui est appelé la composition architecturale.

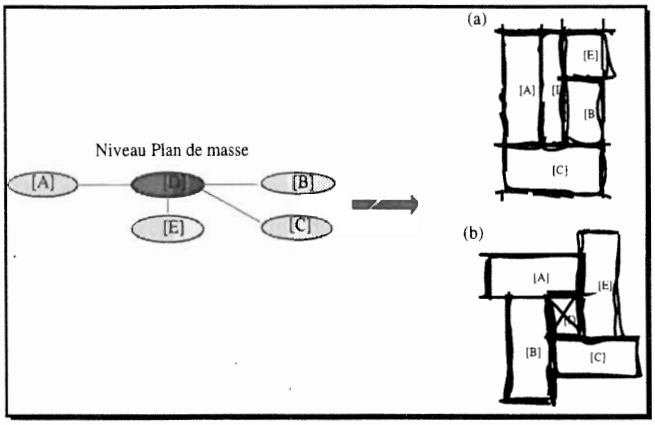


Figure 2.7 : Deux esquisses possibles du *plan de masse*, les arêtes du graphe représentent les adjacences entre les sous-ensembles [A], [B], [C], [D] et [E].

Le mode d'expression utilisé dans cette phase, est essentiellement du dessin *2D* (plans, coupes, façades, etc) dont la particularité est d'être incomplet, imprécis et déformable (l'expression est rarement quantitative). Ce modèle évolutif est un support de réflexion pour l'architecte.

A chaque niveau de conception, le schéma fonctionnel pourra être traduit par plusieurs esquisses. Ainsi, la Figure 2.8 illustre quatre esquisses possibles (S11, S12, S21 et S22) des sous-ensembles [A], [B] et [D] à partir de chaque esquisse du plan de masse proposée dans la Figure 2.7. Il existe donc pour un problème architectural donné un nombre élevé d'esquisses pouvant le satisfaire.

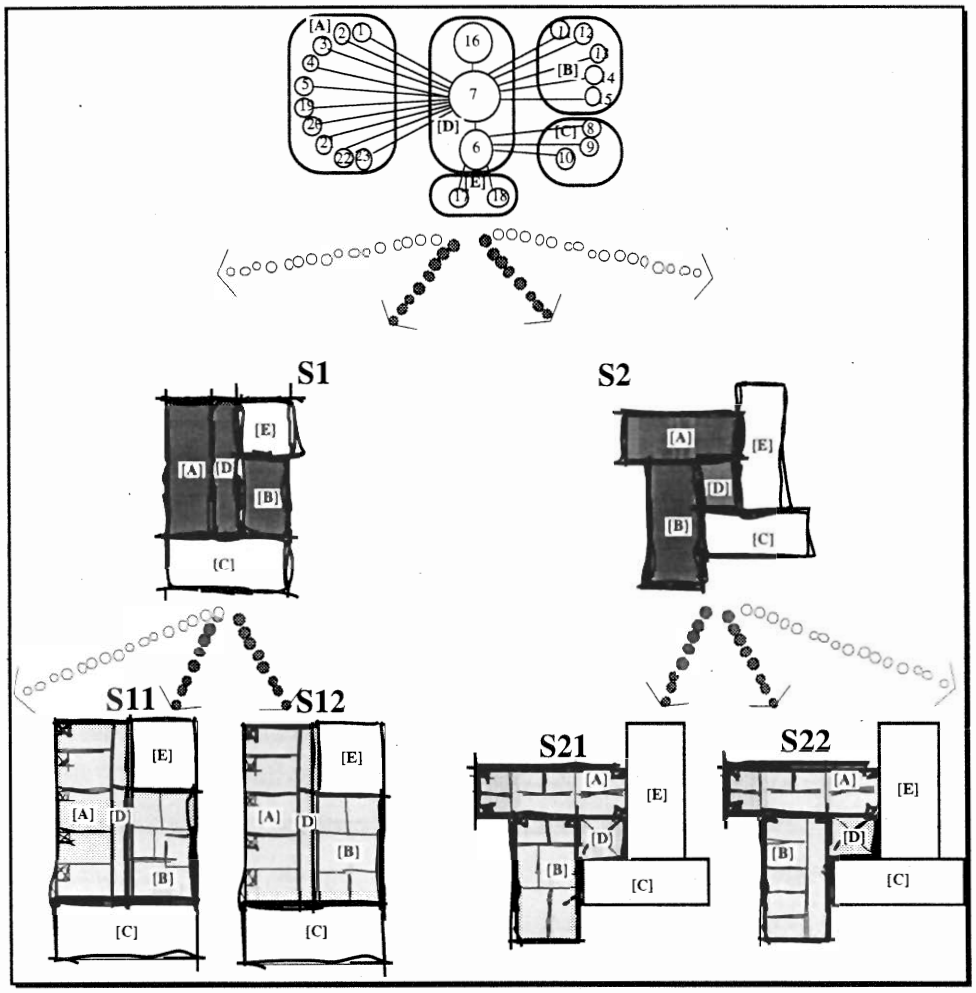


Figure 2.8 : (S1) et (S2) représentent deux esquisses possibles du plan de masse. (S11), (S12), (S21) et (S22) représentent quatre esquisses possibles du sous-ensemble [A]∪[B]∪[D].

On est en droit de se demander ce qui pousse l'architecte à dessiner une esquisse plutôt qu'une autre, ou à n'en dessiner qu'un nombre limité. Est-ce dû à ses propres limites d'imagination ou bien entrevoit-il (comme les joueurs d'échecs) les meilleures solutions immédiatement ? En fait, comment raisonne-t-il pour opérer cette transformation de modèles ? Il serait difficile de répondre à ces questions sans un certain niveau d'abstraction. Il sera difficile de standardiser le processus de raisonnement des architectes en situation de projet car ils ont souvent des méthodes de travail différentes. Par contre, nous allons tenter de dégager les mécanismes les plus importants de manière à mieux comprendre donc à mieux concevoir les outils de CAO du futur.

L'architecte opte pour une forme d'expression plutôt qu'une autre afin d'élaborer son *esquisse* en se référant à son *parti pris*. Par contre, il peut toujours proposer plusieurs solutions. L'homme, contrairement à la machine, est intellectuellement dans

l'impossibilité de considérer d'une manière exhaustive toutes les esquisses. On peut alors, à juste titre, se poser la question si, comme les grand joueurs d'échecs qui s'aident de programmes d'ordinateur pour résoudre des problèmes ardus, les architectes n'auraient pas intérêt à utiliser des logiciels de conception pour être certains de ne pas avoir oublié d'esquisses intéressantes. Le cheminement du schéma fonctionnel à l'esquisse se fait grâce au système d'interprétation des relations topologiques de l'architecte en rapport avec son *parti pris*, qui passe directement d'une relation générale entre les espaces à une solution plus précise. L'architecte crayonne d'ailleurs plusieurs solutions avant de s'arrêter à l'une d'entre elles. Souvent, il commence à dessiner une esquisse qu'il abandonne car, à un moment donné, il ne peut trouver une solution à cause de l'impossibilité de respecter toutes les relations décrites dans le schéma fonctionnel, il va alors procéder à un *retour arrière* sur sa réflexion et reprendre la recherche d'une autre solution.

Dans la Figure 2.9 nous avons illustré un exemple de *retour arrière*. L'architecte va revenir en amont dans l'*historique de conception* et va corriger son erreur en déplaçant un espace. Il n'y a pas de règles prédéfinies qui nous indiquent à quel moment le concepteur s'aperçoit de cette incapacité, chaque architecte réagit différemment. L'architecte va modifier la position d'un espace ; un espace étant placé au nord sera déplacé au sud, tout en restant en conformité avec le schéma fonctionnel. Dans notre cas l'impossibilité de placer le sous-ensemble [E] en relation avec [D] nous a conduit à rebrousser chemin. Cette décision de revenir en arrière s'est produite de manière très naturelle, nous avons pu juger très globalement de l'incapacité d 'aller plus loin. Plus tôt se fait ce constat d'échec, plus nous dirons que l'architecte possède une capacité d'évaluer rapidement la *cohérence topologique de son projet*.

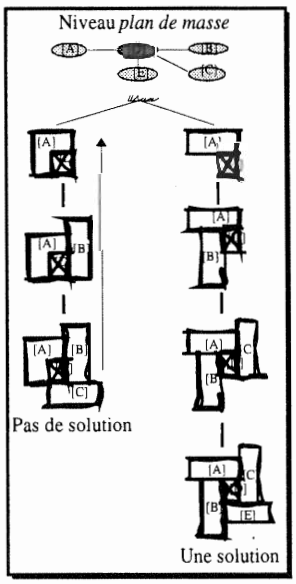


Figure 2.9 : Le *retour arrière* lors de l'élaboration d'une esquisse par un architecte

Les relations entre les espaces représentées dans un schéma fonctionnel sont d'ordre topologique, par exemple une relation d'adjacence entre deux espaces regroupe quatre possibilités (adjacence au nord, au sud, à l'Est, ou à l'ouest). Or l'architecte passe directement du schéma fonctionnel à l'esquisse, ce qui implique dans son raisonnement la recherche d'une relation topologique précise (soit adjacent au nord, au sud, à l'Est ou à l'ouest).

- **La coordination dimensionnelle**

L'étape de coordination dimensionnelle rend faisable le projet du point de vue géométrique (mise à l'échelle) et technique (mise en œuvre du système constructif). Une fois l'élaboration de l'esquisse achevée, l'architecte procède à la mise à l'échelle qui consiste à déterminer avec précision les dimensions des différents espaces. L'architecte a recours pendant cette étape à des éléments de régulation du projet comme le *module*, les *trames* et les *axes*. La mise à l'échelle se fait parallèlement à la mise en œuvre du système constructif choisi (ex : poteaux/poutres, etc), le choix de ce dernier étant fait bien en amont car c'est un critère à prendre en compte dès le départ. Nous pouvons considérer cette phase comme une phase de mise au point numérique des différents paramètres du projet. Le concepteur aura surtout recours à des modification locales d'ordre dimensionnel, contrairement au modifications des relations topologiques plus précises d'une esquisse. Nous avons illustré dans la Figure 2.10 la phase de coordination du projet où nous voyons bien la précision numérique de la géométrie du projet.

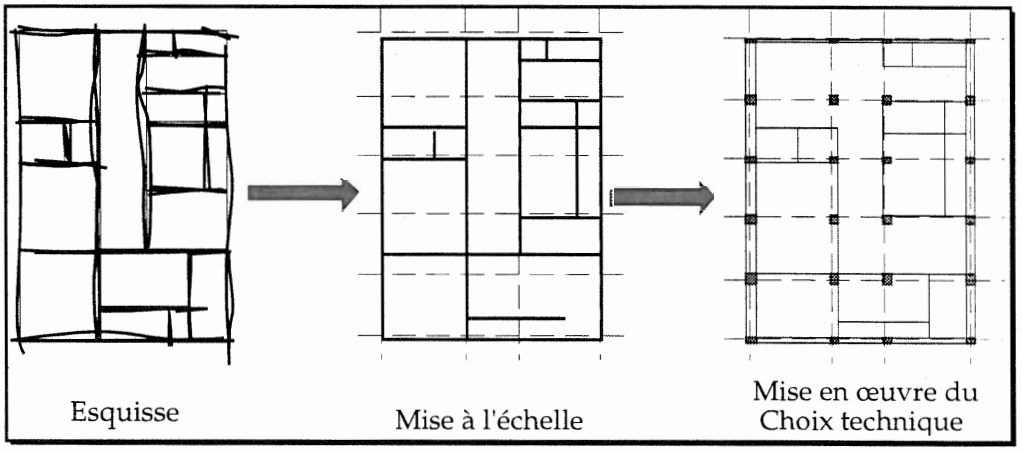


Figure 2.10 : La phase de coordination dimensionnelle : choix d'une structure poteaux/poutres. Le choix d'un autre système constructif aurait probablement donné une autre solution.

2.1.2.3. Notre compréhension du processus de conception

En se basant sur l'analyse précédente, nous pouvons dégager les principales étapes du processus de conception en architecture. Le processus peut être divisé en trois phases :

- **(1) l'établissement du schéma fonctionnel** : élaboration des organigrammes qui décrivent les contraintes spatiales entres les différents espaces qui composent le cahier des charges, cette phase correspond au *niveau fonctionnel d'ARCHiPLAN.*
- **(2) la phase d'esquisse** : c'est la phase de création architecturale, l'architecte élabore des dessins à main levée imprécis et flous, cette phase correspond au *niveau topologique d'ARCHiPLAN.*
- **(3) la coordination dimensionnelle** : c'est l'étape de mise à l'échelle du projet. C'est également là que l'architecte rend constructible son projet, cette phase correspond au *niveau d'optimisation numérique d'ARCHiPLAN.*

  remarque : nous ne nous sommes pas intéressés à la phase d'élaboration du cahier des charges, étant donné que nous le considérons dans ARCHiPLAN comme une donnée d'entrée.

La Figure 2.11 illustre notre compréhension du processus de conception préliminaire.

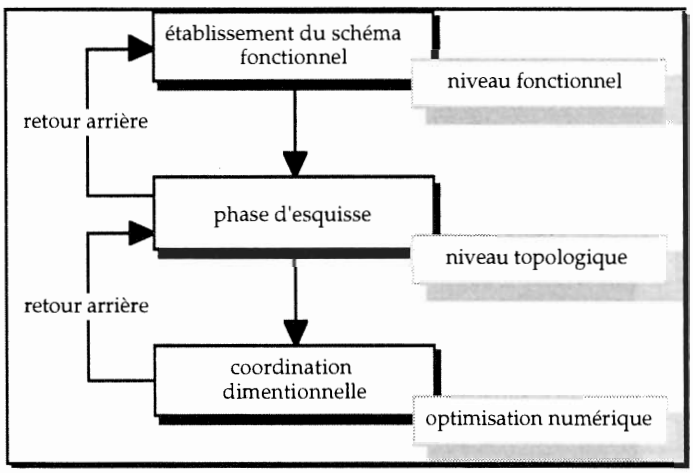


Figure 2.11 : Le processus de conception préliminaire en architecture et le modèle du processus dans ARCHiPLAN

2.2. Les outils informatiques actuels

La naissance de la CAO date du premier logiciel graphique : *Sketchpad* développé au MIT par *Sutherland* en 1963. Les premiers logiciels de CAO du commerce sont apparus sur le marché dans les années 80, grâce à l'amélioration du graphique et de

l'ergonomie des systèmes ainsi que par la diffusion de la micro-informatique individuelle (prix accessibles, d'où une diffusion dans un grand nombre d'agences d'architecture). Du coté des industriels, il était apparu que les techniques graphiques en *2D* et en *3D* étaient les formes d'assistance les plus appropriées à l'activité du dessinateur. L'objectif a été d'enrichir et d'adapter les modeleurs géométriques (exemple : Star, KEOPS, Archicad, Microstation, Arc+, Autocad, etc). L'évolution fut ensuite très rapide grâce à l'apparition des systèmes de gestion de bases de données qui a permis d'étendre les différentes applications au descriptif[17] et au métré[18] (quantitatif).

Aujourd'hui, l'informatique est en train de bouleverser les techniques de la représentation (image de synthèse) mais également les rythmes de la production du projet architectural.

Nous avons représenté dans le Tableau 2.2 les différents outils de CAO correspondant aux différentes techniques informatiques utilisées aujourd'hui. Nous distinguons les prototypes de recherches et les logiciels CAO du marché. Les outils d'aide à la conception préliminaire constitueront très certainement un enjeu important à court terme pour les logiciels CAO du marché.

Techniques Informatiques	Application et stade d'utilisation	Prototypes de Recherche	Outils commerciaux
Systèmes au stade de recherche			
Programmation par contraintes	Aide à la conception préliminaire (phase d'esquisse) aussi bien qu'à l'optimisation de placement ou à la reconception	• ARCHiPLAN,96 ...	
Raisonnement à base de cas (CBR) et programmation par contraintes	Système d'aide à la modification locale ou conception basée sur la détection de cas ressemblants et à l'adaptation	• CADRE, 93 • IDIOM, 94 ...	

17. *Le descriptif : ensemble de documents contractuels dans lesquels un maître d'œuvre décrit les éléments qualitatifs et quantitatifs d'un projet de construction* [DICOBAT, 1991].
18. *Le métré : liste complète détaillée par poste, des travaux correspondant à l'exécution d'un ouvrage, assortie des quantités correspondantes exprimées en mètres linéaires (ml), en m2, en m3 ou en unité (U).* [DICOBAT, 1991].

Systèmes existant sur le marché			
Systèmes à base de connaissances	Projeteur/expert	• Tecton,88 • Krépis,88 ...	• Logiciels spécifiques développés au dessus d'ICAD ...
CAO paramétrique	Projeteur		• PDMS • Microstation ...
Traitements procéduraux (CAO classique)	Descripteur Métreur Dessinateur		• Star • Arc+ • KEOPS • Architrion • Autocad ...

Tableau 2.2 : Les différents outils de CAO selon les techniques informatiques

2.2.1. Les logiciels de CAO commercialisés

Les outils de CAO actuellement sur le marché sont essentiellement orientés vers la représentation et la production du projet architectural, aucune phase préliminaire de conception n'est effectuée à l'aide de ces outils. L'exemple de la conception du centre culturel français de Santiago du Chili [Le Moniteur, 1991], projet remporté sur concours, illustre bien nos propos. Après une phase de réflexion sur les contraintes propres du cahier des charges et les réponses conceptuelles à apporter pour résoudre les problèmes d'allocation spatiale, une première esquisse a été dessinée à la main. Dés que la structure générale du bâtiment s'est avérée satisfaisante, l'architecte l'a saisie sur PC au moyen d'une tablette à numériser. Cette base d'information a permis à l'architecte d'avoir des perspectives afin de cerner parfaitement les volumes intérieurs et extérieurs perceptibles à l'œil. Par ailleurs, cela a permis la production des documents (plans, coupes, façades, etc) pour l'APS[19] (avant projet sommaire) et l'APD (avant projet détaillé).

Comme nous l'avons indiqué dans le tableau 2.2, nous distinguons différents types de logiciels CAO sur le marché :

- (1) **La CAO classique** : dans un système de CAO classique comme Star ou KEOPS, l'utilisateur travaille dans des environnements graphiques et alphanumériques. Le but principal est de construire une maquette numérique

19. L'avant projet : dossier préalable à l'établissement du projet définitif et des plans d'exécution. Un avant projet comporte deux niveaux d'élaboration, qui sont l'Avant Projet Sommaire et l'Avant Projet Détaillé. [DICOBAT, 1991].

qu'on pourra représenter selon différents modes (plans, coupes, perspectives, etc). Ces logiciels disposent en général d'une bibliothèque d'objets prédéfinis (murs, baies, planchers, etc) réutilisables dans plusieurs projets. L'architecte pourra également obtenir son descriptif et son quantitatif sous forme de texte. Tous ces outils sont utilisables seulement quand le projet est conçu, il se situent en aval du processus de conception préliminaire. Les dessinateurs et les métreurs travaillent souvent sur un objet déjà conçu, à un niveau de définition précis sur des tâches d'instrumentation du projet architectural. Les techniques informatiques mises en œuvre sont plutôt des traitements procéduraux, notamment graphiques et de gestion des données. Ces outils sont les plus couramment diffusés dans les agences d'architecture. Actuellement, aucune aide n'est apportée lors de la conception préliminaire d'un bâtiment (phase d'esquisse).

- (2) **la CAO paramétrique** : Contrairement à la CAO classique, la CAO paramétrique a permis au concepteur de tenir quelque peu compte de la conception préliminaire, en permettant de stocker des contraintes géométriques pour pouvoir décliner des géométries avec des dimensions différentes. Il est également possible de définir des relations d'ingénierie avec la géométrie du produit [Galli, 1985 ; Bourdin et Godillot 1991 ; Jung, 1994]. Ces relations sont enregistrées de façon séquentielle comme sur un tableur. Le concepteur peut par exemple définir qu'une droite A est perpendiculaire à une droite B ; à chaque modification de la droite B le système déplace la droite A pour maintenir la relation de perpendicularité. Par contre, la relation n'est pas symétrique, le déplacement de la droite A est impossible. Parmi les systèmes paramétriques utilisés, citons PDMS[20] [Boccon-Gibod et al, 1993].

- (3) **la CAO variationnelle** : Un modeleur variationnel ne présuppose pas d'un ordre de prise en compte des relations d'ingénierie ni d'un sens d'interprétation d'une contrainte géométrique. Ce sont pour lui des équations qu'il résout globalement avec un solveur mathématique [Chung et al, 1989 ; Bourdin et Godillot 1991 ; Paoluzzi, 1992 ; Kurland, 1993]. Contrairement à un modeleur paramétrique, le système variationnel peut prendre en compte et représenter graphiquement des systèmes sous-contraints pour lesquels il existe plusieurs solutions numériques, tout en assurant qu'il existe bien au moins une solution numérique (cohérence).

20. PDMS est une marque déposée de CADCenter (Grande-Bretagne).

2.2.2. La recherche

L'évolution moderne de la CAO s'avère s'appuyer sur des techniques d'Intelligence Artificielle. En effet, il existe actuellement une orientation forte des modeleurs CAO vers les environnements orientés objets et les bases de Données Techniques. C'est dans ce contexte que notre recherche prend place, postulant que le savoir architectural est toujours en friche, et que l'échec de l'algorithmique procédurale classique pourrait être surmonté par de nouvelles techniques de l'IA. Dans ce contexte, de nombreuses recherches ont été entreprises, d'abord dans plusieurs laboratoires Français comme le GAMSAU, sur la modélisation des connaissances en architecture [Hanrot, 1989]. Le groupe LI2a[21] s'est intéressé à la représentation des connaissances architecturales et au problème de l'articulation entre la phase d'esquisse et l'avant-projet sommaire en architecture [Goulette, 1991]. Au CIMA[22], des travaux sur l'apprentissage ont été menés [Guena$_{a-b}$, 1992]. À l'INRIA de Sophia Antipolis, des travaux sur la gestion des contraintes géométriques pour les problèmes d'aménagement spatial ont été entrepris [Charman, 1995]. À l'IRISA-INRIA une méthode de description de scènes architecturales à été développée [Donikian et al 1992]. Dans le cadre d'une étude pour le compte d'EDF en 1992, nous avons défini le cahier des charges d'un système de maintien de cohérence dans une structure de génie civil [Boccon-gibod, 1993 ; Medjdoub et al, 1992]. Dans le cadre des travaux de normalisation, de nombreux projets européens ont été initiés et particulièrement en France par le CSTB [CSTB, 1992]. A l'étranger, nous citerons essentiellement les travaux menés à l'université de Sydney sous la direction de Gero où, parmi les axes essentiels de recherche menés, nous invoquerons le problème de modélisation du processus de conception en architecture fondé sur la logique [Coyne et Gero et al, 1988]. A l'université Carnegie Mellon de Pittsburg aux USA, sous la direction de Flemming, des travaux ont également abordé les problèmes de conception préliminaire en architecture (placement) par l'utilisation de systèmes experts [Flemming, 1988]. A l'École Polytechnique Fédérale de Lausanne, dans le laboratoire d'IA, des travaux sur le maintien de cohérence dans le bâtiment ont été réalisés en utilisant la programmation par contraintes et le raisonnement à base de cas [Hua et Faltings, 1993]. Dans la continuité du projet de Hua et Faltings, Smith a développé un système de composition en architecture nommé IDIOM [Smith, 1996] sur lequel nous allons revenir au chapitre 7.

21. Le Li2a est un laboratoire de recherche à l'École d'architecture de Toulouse.
22. Le CIMA est un laboratoire de recherche de l'école d'architecture de Paris La Villette.

2.2.2.1. De l'intérêt de la synthèse en architecture

Contrairement aux outils d'analyse en architecture qui sont essentiellement dédiés au rendu ou aux différentes phases techniques (structures, ventilation, etc), il est nécessaire d'avoir des outils de synthèse qui permettent de prendre en charge les premières phases de conception. Dans les travaux de recherche qui ont traité des problèmes de placement, différentes techniques ont été utilisées : l'approche "Generate and Test", l'approche dite de "planification de l'aménagement", l'approche optimisation mathématique, l'approche systèmes experts, l'approche programmation par contraintes.

2.2.2.2. L'approche "Generate and Test"

L'approche "Generate and test" consiste à énumérer brutalement tous les n-uplets de solutions de placement et à vérifier a posteriori si ces solutions étaient cohérentes avec les contraintes [Fletcher, 1965]. Cette approche, bien entendu, s'est heurtée au problème de l'explosion combinatoire et au fait que les contraintes du schéma fonctionnel étaient peu extensibles. Cette approche a été totalement revisitée avec les techniques de programmation par contraintes qui, elles, vérifient dès que possible la cohérence des contraintes.

2.2.2.3. L'approche dite de "planification de l'aménagement"

Cette approche consiste à introduire de nouvelles heuristiques (choix d'objet à placer, etc) [Eastman, 1973; Pfefferkorn, 1975]. Ce qui permet d'augmenter l'efficacité de la recherche de solutions par rapport à l'approche précédente.

2.2.2.4. L'approche optimisation mathématique

L'approche optimisation classique [Mitchell et al, 1976 ; Ligett, 1981] a également été utilisée pour résoudre des problèmes d'allocation spatiale. Cette approche s'est heurtée au problème de l'explosion combinatoire et aux contraintes peu extensibles. les solutions sont généralement sous-optimales.

2.2.2.5. L'approche système expert

Les travaux de recherche sur le placement ont repris grâce aux systèmes experts [André, 1986 ; Flemming, 1988 ; Kovacs, 1991]. Kovacs a traité des problèmes d'architecture et André ceux de l'architecture navale. Ces systèmes sont confrontés à

la non-complétude des règles, à l'explosion combinatoire, et trouvent, par nature, des solutions sous-optimales.

2.2.2.6. L'approche programmation par contraintes

Parmi les dernières approches utilisées pour résoudre des problèmes de placement en architecture, nous retrouvons la programmation par contraintes [Baykan et Fox, 1991 ; Maculet, 1991 ; Charman, 1995]. Ces travaux se sont heurtés aux problèmes de l'explosion combinatoire lors de la résolution de problèmes de grande taille.

2.3. Notre cahier des charges

Le cahier des charges d'ARCHiPLAN a été la conséquence d'une analyse du processus de conception préliminaire en architecture et des manquements actuels des logiciels de CAO architecture commercialisés.

2.3.1. Nos hypothèses de travail

Nous avons émis plusieurs hypothèses de départ :

- (1) résoudre des problèmes d'allocation spatiale :
  - dans un premier temps : logement, maison individuelle, bureaux, équipements collectifs (cela fait l'objet de la thèse),
  - dans un second temps : unités de production, services tertiaires (cela fera essentiellement l'objet de propositions dans le chapitre conclusion),
- (2) Aboutir à un outil informatique utilisé par les professionnels :
  - outil graphique type CAO,
  - préférer un dialogue avec le concepteur plutôt que le « tout automatique »,
  - pouvoir traduire la diversité des contraintes en architecture,
  - avoir un système facilement extensible et adaptable.

2.3.2. L'analyse du besoin d'un outil informatique en architecture

2.3.2.1. Les insatisfactions actuelles

Les outils informatiques actuels ne tiennent pas compte de la conception préliminaire en architecture. Les derniers travaux utilisant les systèmes experts se sont heurtés à l'exhaustivité et à la cohérence des règles, à l'obtention de solutions sous-optimales et à des temps de réponses trop importants. Quand aux travaux utilisant la

programmation par contraintes, ils se sont heurtés à l'explosion combinatoire.

2.3.2.2. Les spécifications de notre travail

Les spécifications de notre thèse étaient :

- intégrer la méthodologie de conception architecturale (prendre en compte la phase d'esquisses),
- avoir un outil de conception fonctionnelle (modéliser explicitement le graphe fonctionnel),
- obtenir des solutions optimales de placement au regard de critères d'optimisation explicites et révisables,
- obtenir des temps de réponse acceptables.

2.3.3. Nos premiers choix de travail

Nous avons considéré notre travail sur deux plans : celui du modèle de conception et celui de l'implémentation d'ARCHiPLAN.

2.3.3.1. Nos choix d'un modèle de conception

Dès le départ, nous nous sommes fixés des objectifs que devait atteindre ARCHiPLAN :

- éviter un système type « boîte noire » où le concepteur est relégué au rôle de presse-bouton.
- avoir un outil de CAO qui permette plutôt d'appréhender les solutions interdites afin de délimiter celle qui sont cohérentes et ainsi pouvoir naviguer dans les *champs du possible*, ce qui correspond à :
  - (1) trouver les alternatives de conception,
  - (2) les étudier plus finement,
  - (3) les classer en justifiant leur qualité.

2.3.3.2. Nos choix d'implémentation

À partir du cahier des charges de notre travail, nos choix d'implémentation se sont rapidement portés sur :

- (1) **la programmation orientée objets**, car elle nous permet une extension facile de la base de connaissances architecturales (cf. *chapitre 3).*

- (2) **la programmation par contraintes** (ou « propagation de contraintes ») car elle permet :
  - (1) de résoudre des problèmes fortement combinatoires sur des variables discrètes (technique de cohérence par arcs). Nous considérerons dans nos problèmes de placement des variables entières, ce qui, nous le verrons ne restreint en rien la problématique.
  - (2) d'obtenir des solutions optimales,
  - (3) d'obtenir une première solution en un temps imparti (algorithmes « any time »),
  - (4) de présenter une grande déclarativité et un découplage total entre la déclaration des contraintes et les algorithmes de résolution, ce qui permet :
    - (1) une extension aisée des contraintes,
    - (2) une diversité des contraintes fonctionnelles,
  - (5) la possibilité d'introduire des heuristiques dynamiques dans les algorithmes de résolution, ce qui permet un rapprochement avec le raisonnement du concepteur.

2.4. Conclusion3

Dans ce chapitre, après avoir évoqué les différentes acceptions de l'architecture, analysé le processus de conception et après avoir examiné les travaux de recherche faits dans le domaine, nous avons défini le cahier des charges d'ARCHiPLAN, et étayé nos principaux choix d'implémentation : la programmation orientée objets et la programmation par contraintes.

Chapitre 3

Le modèle de connaissances architecturales

Nous présentons ici le modèle de connaissances d'ARCHiPLAN. Il se compose de deux parties : les espaces architecturaux manipulés, et les contraintes spatiales qui imposent des relations entre ces espaces. Par rapport au travaux précédents sur le placement, nous apportons des enrichissements sur plusieurs plans :

- (1) Au niveau de la structure des objets proposés où nous avons tenu compte des espaces de circulation verticaux (escaliers). Cela nous permettra, par la suite, d'être les premiers à synthétiser des solutions sur plusieurs étages.
- (2) Au niveau des contraintes qui ont été élaborées dans l'objectif de tenir compte de la méthodologie de conception préliminaire en architecture (i.e. l'introduction de la phase d'esquisse), et au niveau de la proposition de nouvelles contraintes efficaces pour la réduction de l'espace de recherche.

3.1. Le modèle des espaces architecturaux

3.1.1. Les modèles existants

3.1.1.1. La structuration des objets

La modélisation des connaissances architecturales a fait l'objet de plusieurs travaux de recherches. Le *GSD* « *Groupe de Structuration des Données* » [GSD, 1991], regroupant plusieurs équipes de recherche françaises, a défini plusieurs modèles de connaissances du bâtiment, la méthode NIAM[23] ayant été utilisée comme formalisme. L'ensemble des modèles proposés illustre la démarche qui consiste à passer progressivement de concepts génériques reconnus par tous à des concepts de plus en plus spécialisés orientés applicatifs (thermique, énergétique, etc.). Ces travaux ont permis de s'accorder sur un vocabulaire, un formalisme commun et sur les limites du domaine dans lequel la synthèse est effectuée. Cette synthèse a été indispensable pour élaborer un langage commun pour la construction.

Dans le cadre de la modélisation des connaissances architecturales, nous citerons également le travail de [Hanrot, 1989] dans lequel un langage pour la conception des bâtiments est défini et un *corpus architectural* est proposé à travers une taxinomie par spécialisation et par partition. Hanrot a choisi la méthode *KOD* « *Knowledge Oriented Design* » comme formalisme. Comme il est illustré dans la Figure 3.1 le *corpus architectural* représente les objets architecturaux spécialisés en cinq sous-classes.

23. NIAM (Nijssen Information Analysis Methode) est un formalisme pour traduire le concept d'entités-relations ou de réseaux sémantiques, c'est-à-dire d'objets reliés par des relations.

Objets-A	Ensemble-Ax	ex: ensemble-historique, grand ensemble...
	Entités-Al	ex: hôpital, immeuble d'habitation...
	Divisions-Al	ex: escalier, étage, logement, pièce...
	Eléments-Ax	ex: baie, mur de refend, pilier, bidet...
	Constituants-Ax	ex: bloc, profilé, brique, vitrag...
	Partie architecturale	ex: pan, corps...

Figure 3. 1 : *Le Corpus Architectural* [Hanrot, 1989]

Hanrot a également mis en rapport les différent niveaux composant le *corpus architectural*, l'échelle et le type de dossier correspondant (cf. Figure 3.2). ARCHiPLAN se situera au niveau des *divisions*, c'est-à-dire manipulera des étages et des pièces.

Niveaux de définition/ Objets architecturaux	Echelles	Type de dossier	Projet niveaux
Ensembles (parc, lotissement...)	1/1000, 1/500	Urbain	
Entité (édifice, bâtiment, hôpital...)	1/500, 1/200	Esquisse/ Projet sommaire	
Division (étage, pièce, structure...)	1/200, 1/100	Projet sommaire	ARCHiPLAN Maculet Charman
Elément (mur, baie...)	1/100, 1/50	Projet détaillé/ Projet sommaire	
Constituant (briques, profilés...)	1/50, 1/20	Projet exécution Projet détaillé	

Figure 3.2 : Les niveaux de définition d'après [Hanrot, 1989]

Baykan et *Fox* [Baykan et Fox, 1991] ont abordé le problème d'aménagement d'une pièce. Ils ont proposé un modèle de connaissances définissant les principaux éléments pour la conception d'une cuisine (cf. Figure 3.3). Au niveau conceptuel ils ont présenté une hiérarchie de classes, mais au niveau de l'implémentation informatique, seul l'identificateur des instances change (cuisine, circulation...), c'est-à-dire qu'il n'existe pas de méthodes spécifiques aux différentes classes. Tous les éléments sont issus de la classe rectangle (*design unit*).

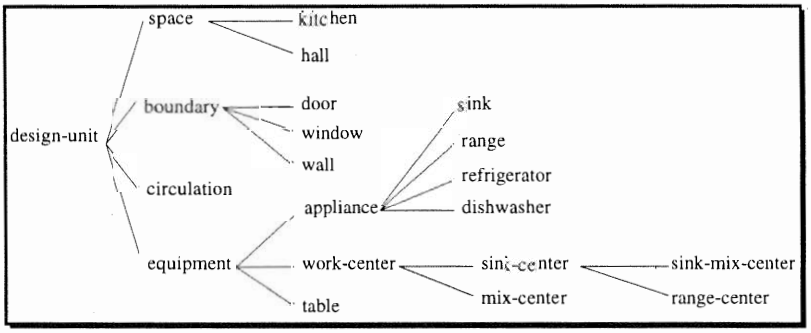


Figure 3.3 : Modèle de connaissances proposé par Baykan et Fox [Baykan et Fox, 1991]

3.1.1.2. La modélisation géométrique des objets

Différents modèles de représentation géométrique ont été proposés. Ainsi une armoire est représentée par un parallélépipède ou un rectangle, une mosaïque est représentée par un polygone... La variété des différents modèles géométriques porte essentiellement sur trois variables [Maroy, 1973 ; Roach, 1984 ; Mukerjee et al, 1991] :

- les variables de position,
- les variables d'orientation,
- les variables de dimension.

• La modélisation discrète

Cette modélisation consiste à partitioner l'espace en cellules, souvent rectangulaires ou parallélépipédiques, de taille fixe ou variable. Chaque cellule est, soit libre, soit occupée par un espace. Cette disrétisation peut être à pas fixe [Eastman, 1973] ou à pas variable [Charman, 1995].

• La modélisation semi-continue

Dans ce type de modèle, certaines variables géométriques sont continues, d'autres discrètes. L'un des cas les plus étudiés est le rectangle isothétique[24], *Maculet* [Maculet, 1991] a défini une algèbre sur ces rectangles appelée *algèbre de Manhattan*. Il a généralisé son modèle à un modèle *3D* appelé *boîtes de Manhattan*[25] (cf. Figure 3.4), la conception est une composition de ces boîtes. L'utilisation du *3D* est indispensable dans certains cas comme l'industrie nucléaire ou chimique, mais son utilisation reste marginale par rapport au *2D* ou *2D 1/2*.

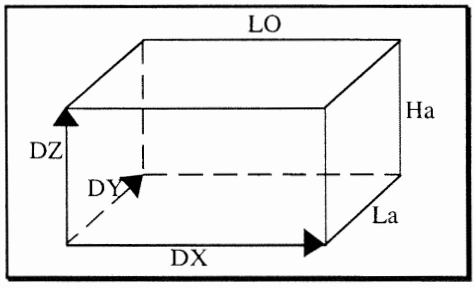


Figure 3.4 : Boîte de Manhattan [Maculet, 1991]

24. Un rectangle isothétique est un rectangle dont les côtés sont parallèles aux axes ox et oy [Charman, 1995].
25. Une boîte de Manhattan est un parallélépipède rectangle, dont les faces sont parallèles aux plans du repère absolu.

- **La modélisation continue**

Cette représentation revient à ce que les variables géométriques des différents objets appartiennent à des domaines continus. La modélisation la plus simple des objets est une approximation parallélépipédique en *2D* avec des possibilités de rotation par rapport aux axes principaux de l'*espace de placement*. L'inconvénient majeur de cette représentation reste la complexité du problème où la résolution demeure impraticable dès que le nombre d'objets est important [Pimont et al, 1993].

3.1.2. Notre modèle des espaces architecturaux

3.1.2.1. Notre analyse et nos choix

Aucun modèle proposé par le *GSD* n'a été élaboré pour la phase d'allocation spatiale en architecture. Tous se situent en aval de cette étape. Les modèles proposés par *Maculet, Baykan* et *Charman* ont, par contre, été dédiés à des problèmes de placement, mais aucun de ces modèles n'a tenu compte des circulations verticales afin de résoudre l'allocation spatiale d'un bâtiment à plusieurs étages. De plus, mais ceci ne concerne pas ce chapitre, tous ces modèles sont désespérément confrontés à l'explosion combinatoire et donc non utilisables en pratique car les seules solutions sont d'ordre numérique (cf. *chapitre 4*).

Notre démarche pour définir le modèle de connaissances d'ARCHiPLAN repose sur quatre aspects fondamentaux :

- **(1) aspect méthodologique** : nous avons tenu compte des principales phases de conception préliminaire en architecture. Outre les résultats numériques que nous avons obtenus, notre souci est d'intégrer une phase intermédiaire qui corresponde à la phase d'esquisse (cf. notre définition d'une solution topologique au *chapitre 4*). La modélisation des contraintes et essentiellement des contraintes d'adjacence et de non-recouvrement entre les espaces a été élaborée dans l'objectif que les solutions topologiques aient la propriété d'être des classes d'équivalence distinctes de solutions numériques. Nous détaillerons cet aspect au *chapitre 4*.
- **(2) aspect applicatif** : nous avons tenu à étendre notre application à des bâtiments à plusieurs étages. Les circulations verticales (escaliers) font bien partie du modèle de connaissances d'ARCHiPLAN.
- **(3) aspect géométrique** : nous avons opté pour une modélisation discrète des espaces architecturaux. Ce choix est justifié par l'application même. En effet, en

architecture, l'unité géométrique de base est le *module*. Tous les attributs géométriques des classes que nous avons défini sont donc également des variables contraintes entières.

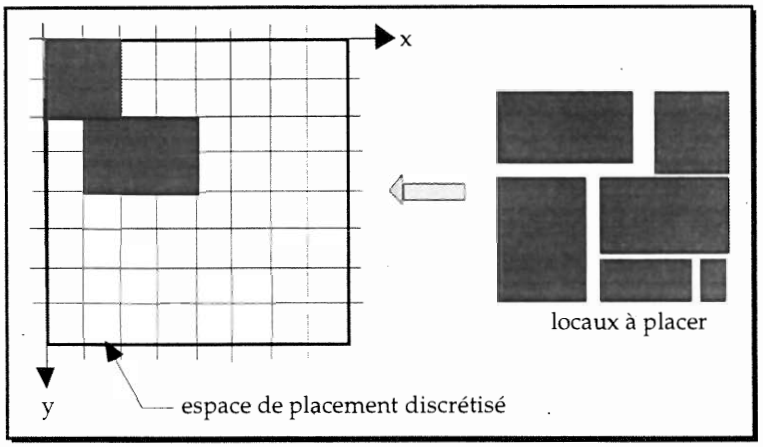


Figure 3.5 : Notre représentation géométrique des espaces

- **(4) Aspect technique** : découplage des algorithmes d'énumération, des espaces architecturaux et des contraintes spatiales. Afin de permettre une extensibilité de la base de connaissances et la mise en œuvre du niveau topologique, nous avons utilisé l'approche orientée objets [Coad, 1991 et 1992] associée à une technique de résolution par propagation de contraintes (*Annexe A.1.*) et plus particulièrement la *cohérence par arcs* sur les entiers.

3.1.2.2. Les classes des espaces architecturaux

Notre modèle de connaissances (cf. Figure 3.6) regroupe les principaux éléments architecturaux correspondant aux vides (les éléments de structure ne sont pas pris en compte).

Nous avons utilisé un modèle de représentation objets pour représenter les espaces architecturaux. Chaque classe est définie par des attributs et des méthodes spécifiques. Le mécanisme d'héritage permet d'hériter des attributs et des méthodes des classes supérieures. Le mécanisme d'instanciation permet de créer des objets réels avec des valeurs spécifiques. La seule différence de notre modèle par rapport à une classification objets classique est la possibilité de définir des contraintes de classe[26] entre des attributs contraintes d'une classe. La classe la plus générale est la classe *espace*. Il a été spécifié trois principales sous-classes : les locaux, les circulations et les étages. Le modèle de connaissances est extensible à d'autre classes.

26. Une contrainte de classe est une spécificité de la librairie de programmation par contraintes PECOS sur les objets de Le-Lisp très intéressante pour des modèles de produit. C'est une contrainte générique, c'est-à-dire qu'elle est partagée par toutes les instances d'une classe donnée. Grâce au mécanisme d'héritage, les contraintes de classe sont héritées par toutes les classes descendantes d'une classe donnée [Ilog, 1991].

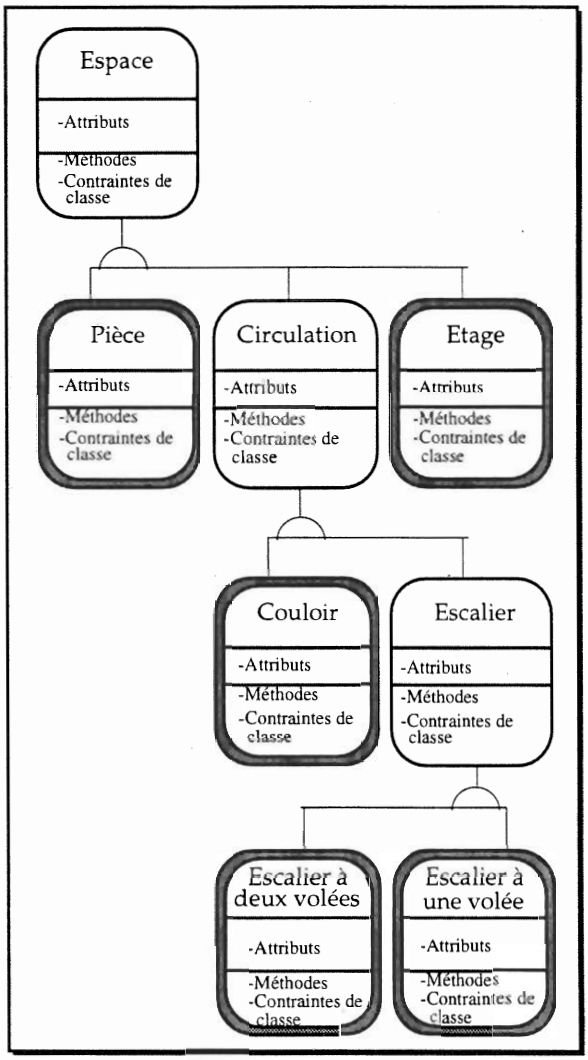


Figure 3.6 : Modèle de représentation des classes d'objets architecturaux

• **La classe espace**

La classe *espace* est une classe abstraite[27] qui représente, dans notre contexte, un rectangle. Elle permet de factoriser tous les attributs, toutes les contraintes de classes, et toutes les méthodes relatives aux rectangles. Cette classe est caractérisée par un *identificateur*, deux *points de référence* (*x1*, *y1*) et (*x2*, *y2*), une *longueur* (*L*), une *largeur* (*W*), une *surface* (*S*), un coefficient appelé *degré de contraintes* (*dg-cont*) et une *liste d'identificateurs d'espaces adjacents* (*liste-esp-adj*). L'*identificateur* est une chaîne de caractères indiquant le nom d'une instance de cette classe, c'est-à-dire le nom donné à un espace (qui correspond à sa fonction). Les deux *points de référence* la *longueur*, la *largeur* et la *surface* sont des variables entières contraintes. Le *degré de contraintes* est une valeur entière dynamique. La liste des *identificateurs* est un ensemble des *identificateurs* d'instances. Le *degré de contraintes* et la *liste d'identificateurs* des espaces

27. Une classe abstraite est une classe représentant une abstraction sans correspondance avec un objet physique.

adjacents seront utilisés lors de la recherche de solutions (cf. *chapitre 4*). Comme indiqué dans la Figure 3.7, la classe *espace* est représentée géométriquement par un rectangle isothétique. Cette représentation sera reprise par toutes les classes définies à partir de la classe *espace*. Une des méthodes de la classe *espace* est la représentation écran d'un *espace* qui sera abordée au *chapitre 5*.

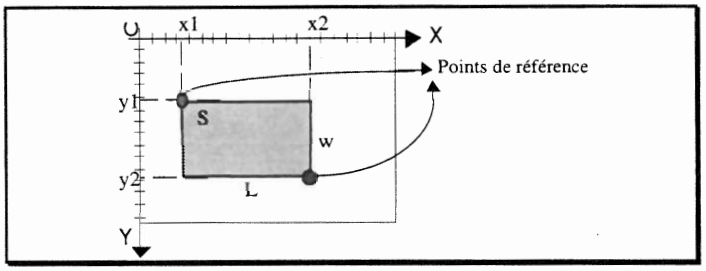


Figure 3.7 : Représentation géométrique de la classe *espace*

- **Attributs de la classe espace** :
  - Identificateur
  - $x1 \in [x_{1a} \ldots . x_{1b}]$
  - $y1 \in [y_{1a} \ldots . y_{1b}]$
  - $x2 \in [x_{2a} \ldots . x_{2b}]$
  - $y2 \in [y_{2a} \ldots . y_{2b}]$

  } ⇒ Variables contraintes et leur domaine initial

  - $L \in [1 \ldots . l_b]$
  - $W \in [1 \ldots . w_b]$
  - $S \in [1 \ldots . s_b]$
  - Degré de contraintes (dg-cont)
  - Liste d'identificateurs des espaces adjacents (liste-esp-adj)
- **Contraintes de classes de la classe espace** : nous avons défini trois contraintes de classe afin d'assurer la cohérence géométrique de cette classe :
  - (c1) $x2 = x1 + L$
  - (c2) $y2 = y1 + W$
  - (c3) $S = L \times W$

La modification d'une variable composant la contrainte (c1), (c2) ou (c3) entraîne la modification des domaines des variables qui lui sont associées, grâce à la *cohérence par arcs* sur les entiers (*Annexe A.1.*) que nous avons utilisée. Ces trois contraintes seront toujours respectées. La Figure 3.8 illustre la *cohérence par arcs* sur les entiers, à travers l'exemple de la pose d'un espace *e1* dans un *espace de placement* avec des dimensions fixées à [0,10]x[0,10] tel que $e1.L \in [2,6]$, $e1.W \in [2,6]$, $e1.x1 \geq 0$, $e1.y1 \geq 0$, $e1.x2 \leq 10$ et $e2.y2 \leq 10$. Une contrainte supplémentaire sur la surface : $e1.S > 12$ va déclencher une réduction des domaines de *e1.L* et de *e1.W* à [3,6] puis, par contre-coup, une réduction des domaines de *e1.x1*, *e1.y1*, *e1.x2* et *e1.y2*. Cette réduction des domaines de *e1. L* et

e1.W est caractéristique de la propriété de cohérence par arcs (cf. *annexe* a.1). En effet, la valeur $e1.L_i$=2 ne peut pas être conservée dans le domaine de définition de *e1.L* car il n'existe aucune valeur $e1.W_i$ de *e1.W* appartenant à l'intervalle [3,6] telle que $e1.S=e1.L_i\ e1.W_i \in [13,36]$. En effet, pour $e1.L_i$=2, on n'a plus *e1.S*=12 pour $e1.W_i$=6. Le raisonnement est identique pour $e1.W_i$=2.

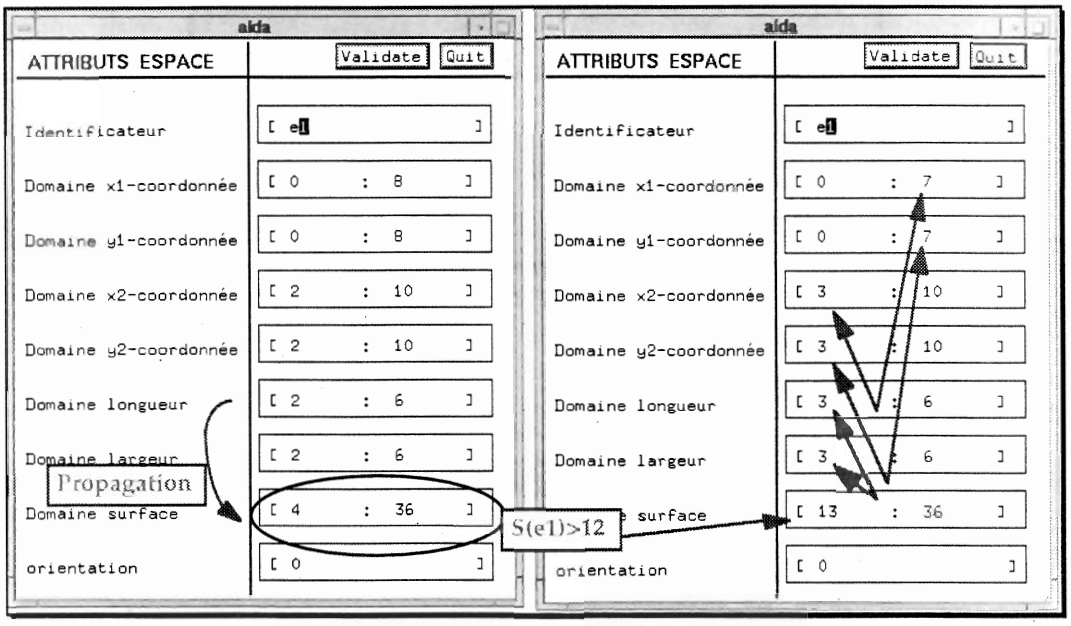


Figure 3.8 : Illustration de la méthode de cohérence par arcs

- **La classe pièce**

La classe *pièce* définit les espaces autres que les circulations. Elle hérite, bien entendu, de tous les attributs, méthodes et contraintes de classe de la classe *espace*. Cependant, elle est caractérisée par un attribut qui est l'*orientation*. Cet attribut d'orientation possède un domaine à deux valeurs *0°* et *90°*. En effet, en disant, par exemple, que nous désirons qu'une pièce ait un côté compris entre *2* et *6* mètres et l'autre entre *2* et *4* mètres, nous ne référençons particulièrement ni la longueur *L*, ni la largeur *W*. Il faut donc en fait considérer les deux configurations possibles (cf. Figure 3.9) grâce à l'orientation. Pour ces deux sous-problèmes contraints, il apparaît des solutions identiques lors de leur énumération ; par exemple : si à *0°* l'une des solutions possibles est une longueur de *3* mètres et une largeur de *2* mètres et à *90°* l'une des solutions possibles est une longueur de *2* mètres et une largeur de *3* mètres, dans ce cas nous avons deux solutions redondantes (cf. Figure 3.10). Pour cela nous avons développé une contrainte de classe d'*élimination des redondances d'orientation*.

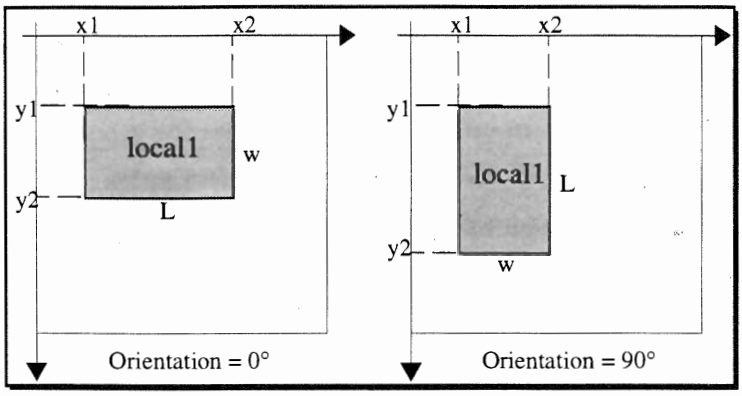


Figure 3.9 : Configurations géométriques de la *pièce* local1 pour chaque orientation

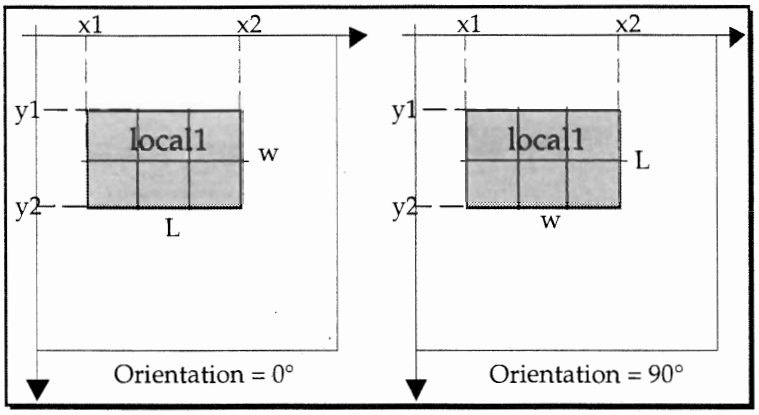
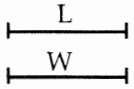
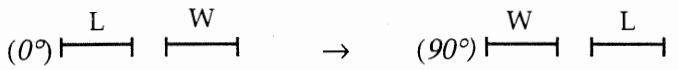


Figure 3.10 : Exemple de redondance de la *pièce* local1

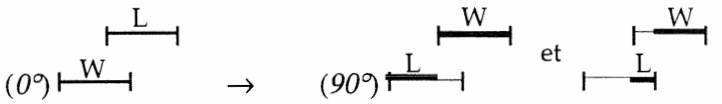
- **Attributs de la classe pièce** :
  - Orientation ∈ {0° 90°}
- **Contrainte de classe d'*élimination des redondances d'orientation* d'une pièce** : cette contrainte est activée à chaque instanciation de la classe *pièce*. Elle est propagée dès que l'*orientation* de la *pièce* est instanciée. Nous comptons plusieurs cas de figure dépendant des domaines de *L* et de *W* :
  - **cas 1** : si les domaines de la *longueur* et de la *largeur* d'un pièce sont égaux, avant même l'énumération de l'orientation de cette pièce, l'attribut d'orientation verra son domaine réduit à la valeur de *0°*, car à *90°* toutes les solutions seront redondantes.



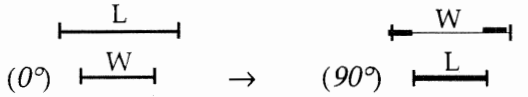
  - **cas 2** : si la valeur maximale du premier côté est inférieure à la valeur minimale du deuxième côté, alors il n'y a pas de solution redondante à 90°.



- **cas 3** : si les domaines initiaux de la *longueur* et de la *largeur* s'intersectent sans être inclus l'un dans l'autre, alors à *90°* il faut considérer deux sous-problèmes pour ne pas avoir de redondance avec l'orientation 0° :



- **cas 4** : si le domaine d'un côté est inclu dans celui du deuxième côté, on considérera à 90° le sous-problème suivant :



Nous avons illustré en *annexe 2.1* la contrainte d'*élimination des redondances* où chaque cas correspond à l'activation d'une contrainte. Cette contrainte passe par un mécanisme d'événement de propagation associé à l'instanciation de la variable *orientation* de la *pièce* ; c'est un mécanisme de *démon* correspondant à une activation conditionnelle.

[Charman, 1995] a également proposé une contrainte d'*élimination des redondances* mais nous avons constaté qu'elle n'est pas satisfaisante car elle n'élimine pas toutes les redondances.

• La classe circulation

La classe *circulation* est une classe abstraite. Elle ne comporte aucun attribut ou contrainte de classe de plus que la classe *espace*. Le rôle de la classe *circulation* se situe essentiellement au niveau de la prise en compte particulière des circulations dans les algorithmes de placement sur plusieurs étages. On placera en fait les circulations en premier pour propager des conséquences importantes sur tous les étages. A partir de cette classe, d'autre sous-classes ont été définies regroupant la circulation verticale (couloir, hall...) et la circulation horizontale (escalier, monte-charge, ascenseur...).

• La classe couloir

La classe couloir possède les mêmes attributs et contraintes de classe que la classe pièce. Cette classe a été définie pour bien souligner la différence fonctionnelle entre un pièce (pièce d'habitation, local technique, local de travail...) et une circulation (couloir, hall...). Cette différence prendra toute son importance lors de la pose des contraintes géométriques entre les différents espaces. Ainsi une relation d'adjacence entre une pièce et un couloir se comporte différemment d'une relation entre deux couloirs, cette différence est gérée grâce à l'existence de ces deux classes bien distinctes.

• La classe escalier

La classe *escalier* est une classe abstraite. Cette classe regroupe tous les attributs et contraintes de classe communs à tous les types d'escaliers. Un escalier est caractérisé par un *point de référence* de la première marche ou *marche de départ* (*x-marche-de-départ y-marche-de-départ*), un point de référence de la dernière marche ou *marche d'arrivée* (*x-marche-d'arrivée, y-marche-d'arrivée)*, la longueur des marches (*l-marche*) et l'*orientation* qui peut avoir quatre valeurs possibles *0°, 90°, 180°* et *270°*. Les *points de référence* et la *longueur* de l'escalier sont des variables contraintes entières.

- **Attributs de la classe escalier :**
  - $\text{Orientation} \in \{0°, 90°, 180°, 270°\}$
  - $x - \text{marche} - \text{de} - \text{départ} \in [x_{da} \ldots . x_{db}]$
  - $y - \text{marche} - \text{de} - \text{départ} \in [y_{da} \ldots . y_{db}]$
  - $x - \text{marche} - \text{d'arrivé} \in [x_{aa} \ldots . x_{ab}]$
  - $y - \text{marche} - \text{d'arrivé} \in [y_{aa} \ldots . y_{ab}]$
  - $L - \text{marche} \in [1 \ldots . L_b]$

  $\Rightarrow$ Variables contraintes

• L'escalier à une volée

L'escalier à une volée (cf. Figure 3.11) est une spécialisation de la classe *escalier*. Il est caractérisé par la contrainte de classe qui permet de maintenir sa cohérence géométrique et fonctionnelle à chaque instanciation de la variable orientation. Comme indiqué dans la Figure 3.12, cet escalier peut avoir quatre configurations géométriques possibles.

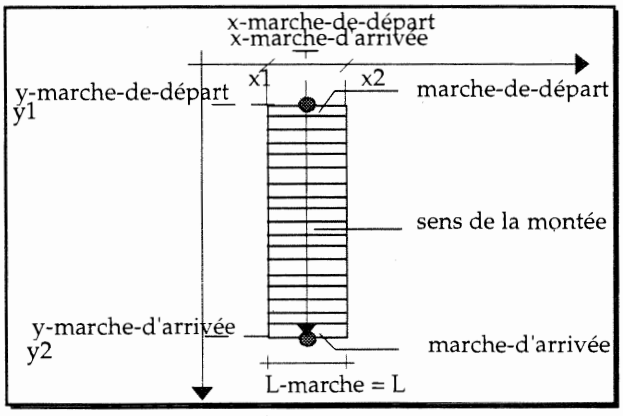


Figure 3.11 : Représentation géométrique d'un escalier à une volée

- **Contraintes de classe de l'escalier à une volée** : elle assure la cohérence des quatre configurations géométriques correspondant à chacune des instances de la variable orientation (voir Figure 3.12).

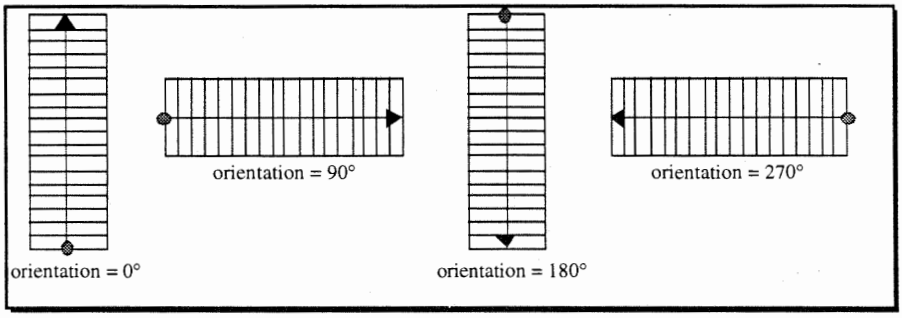


Figure 3.12 : Les quatre orientations possibles d'un escalier à une volée

Les méthodes de propagation déclenchées à chaque instanciation de l'orientation sont indiquées en *annexe 2.2*. Chaque cas de cette contrainte "démon" assure la cohérence géométrique entre les marches de départ et d'arrivée, et les attributs de bases hérités de la classe *espace*.

- **L'escalier à deux volées**

L'*escalier à deux volées* (cf. Figure 3.13) est une spécialisation de la classe *escalier*. Cette classe est caractérisée par un attribut représentant la *position* de la *marche de départ*. Cet attribut est une variable contrainte possédant deux valeurs : *gauche* et *droite*.

- **Attribut de la classe escalier à deux volées** :
  - Position-marche-depart∈ {gauche, droite}

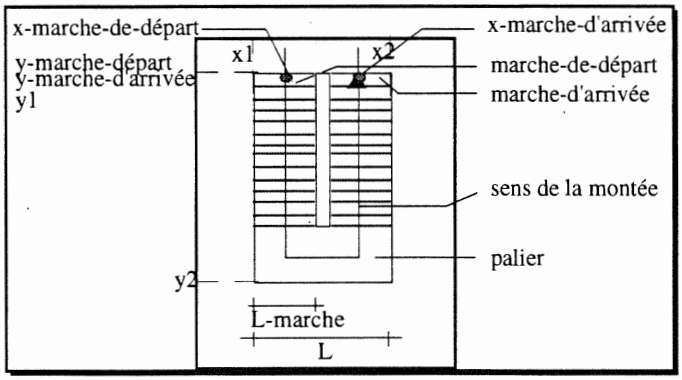


Figure 3.13 : Représentation géométrique d'un escalier à deux volées

- **Contrainte de classe de l'escalier à deux volées :** Comme l'escalier à une volée, l'escalier à deux volées possède une contrainte de classe lui permettant de maintenir sa cohérence géométrique et fonctionnelle (elle est donnée en *annexe 2.4*). Cette contrainte définit 8 sous-problèmes contraints. La contrainte démon est activée lorsque les variables *orientation* et *position-marche-de-départ* sont instanciées. Chaque sous-problème ou choix correspond à une configuration géométrique indiquée dans la Figure 3.14.

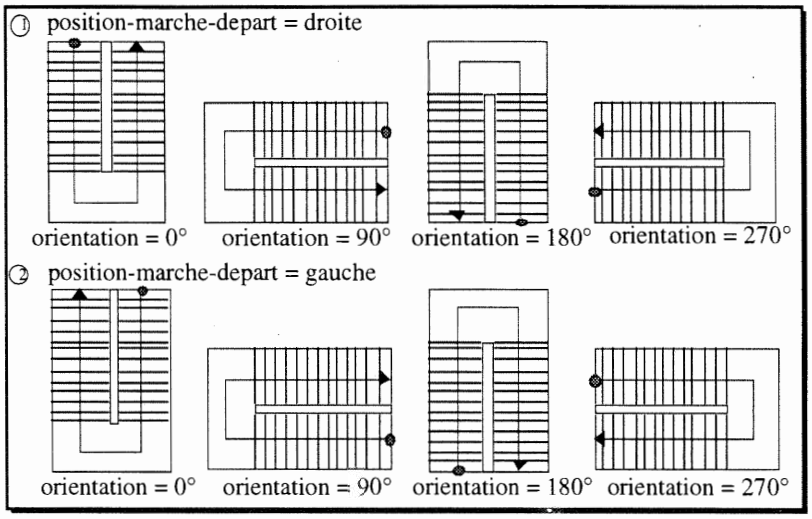


Figure 3.14 : Les huit orientations possibles d'un escalier à deux volées

- **La classe étage**

La classe *étage* correspond à l'*espace de placement*, c'est-à-dire au contour dans lequel on place les pièces. Dans un premier temps cet *espace de placement* est limité à un rectangle. Nous verrons comment lever partiellement cette contrainte au *chapitre 7*. Chaque *espace de placement* est une instance de cette classe. Tous les espaces définis dans un cahier des charges sont positionnés dans un étage. Dans les chapitres qui vont suivre, la terminologie *espace de placement* sera la plus utilisée. La classe étage

peut être spécialisée à d'autres sous-classes comme une classe *Rez de chaussée, étage courant*... Dans notre modèle, nous nous contentons de la seule classe Étage. Pour définir une instance correspondant à un *rez de chaussée* ou à un *premier étage* il suffit de l'indiquer à partir de l'identificateur de l'instance. Les étages ne sont pas différenciés.

L'étage a les mêmes attributs que la classe *espace*. Dans la plupart des cas de conception de logements familiaux ou autres, les attributs *L*, *W* et *S* des espaces de placement sont imposés, leurs domaines sont donc instanciés. Bien que ce soit le cas, notre approche permet de considérer les attributs d'un *espace de placement* comme ceux d'un espace, c'est-à-dire définis sur un domaine. Les étages sont reliés entre eux par la contrainte *Sur*. Ils constituent les seules circulations verticales (ex: ascenseur, escaliers, gaine...) qui assurent la communication entre étages. Nous proposons au *chapitre 4* un algorithme d'énumération de solutions topologiques et numériques qui opère par parties sur les étages.

3.2. Le modèle des contraintes spatiales

3.2.1. Les modèles existants

Dans les travaux de recherche précédents et actuels sur l'allocation spatiale (placement) en architecture, des modèles de représentation de contraintes ont été proposés.

Maculet [Maculet, 1991] a défini un tableau de relations spatiales qui permettent de traduire le cahier des charges d'un bâtiment. Ce modèle est constitué d'une bibliothèque de contraintes géométriques indiquées dans la Figure 3.15. Ces relations géométriques sont beaucoup trop élémentaires pour permettent de décrire dans la pratique d'un architecte le cahier des charges d'un bâtiment.

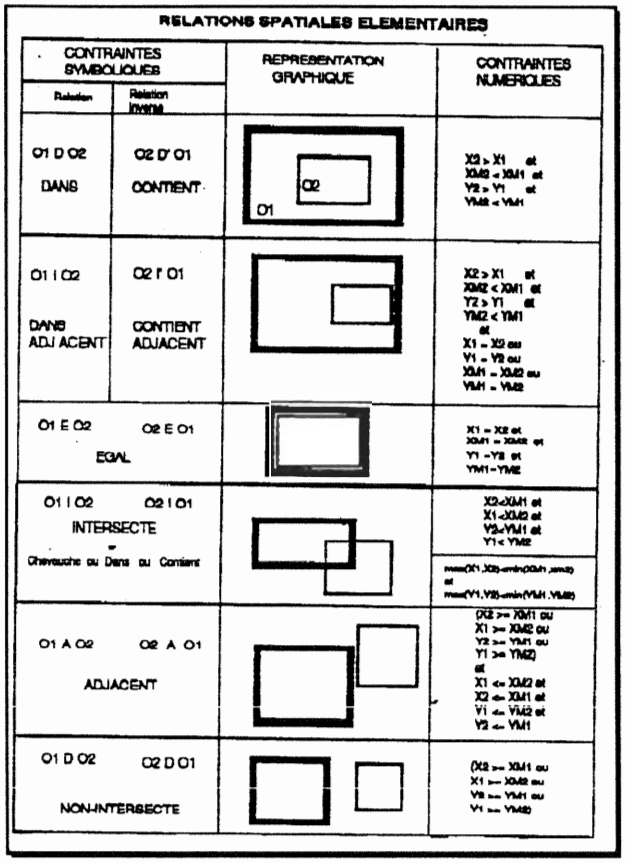


Figure 3.15 : Les relations spatiales élémentaires de Maculet [Maculet, 1991]

Charman [Charman, 1995], quant à lui, a proposé deux types de contraintes :

- (1) **les contraintes implicites qui regroupent :**
  - (1) les contraintes d'inclusion, qui imposent aux espaces d'être dans l'*espace de placement*,

- (2) les contraintes de non-recouvrement qui interdisent aux espaces de se recouvrir,
- (3) la contrainte qui impose le recouvrement total de l'*espace de placement*,
- (4) la contrainte de symétrie, qui permet de détecter les symétries entre deux espaces (ex : deux chambres identiques) et de les éliminer.

- (2) **les contraintes spécifiques** : ces contraintes correspondent aux spécifications du problème (contraintes d'adjacence...),

Afin d'améliorer les performances Charman [Charman, 1995] propose d'autres contraintes qui sont redondantes avec les contraintes implicites mais permettent d'accélérer les propagations :

- (1) **les contraintes de partition** : Lors de l'énumération des solutions numériques de placement, les espaces sont pris un à un (dans un ordre réfléchi) et « placés » dans l'*espace de placement* à un endroit donné et avec des dimensions données. La contrainte de partition permet, en cours de placement, de détecter des espaces libres incohérents (cf. Figure 3.16), c'est-à-dire des endroits non encore utilisés mais qui sont trop étroits pour placer un des quelconques espaces restants.

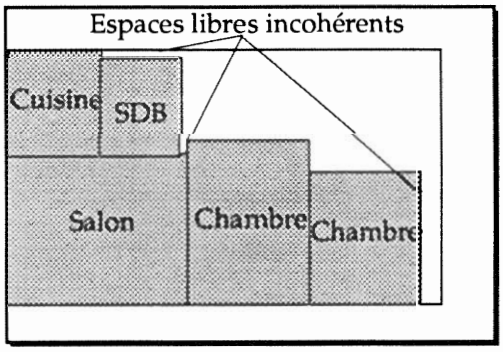


Figure 3.16 : Les espaces libres incohérents selon *Charman* [Charman, 1995]

- (2) **la contrainte du nombre de rectangles** : elle vérifie en cours de placement que le nombre minimal de rectangles dans l'espace libre restant à remplir est inférieur ou égal au nombre d'espaces restant à placer.

Outre les nouvelles contraintes qu'il a proposé, *Charman* a adapté la formulation d'un *CSP « Constraint Satisfaction Problem »* (cf. *annexe A.1.*) à un *problème de satisfaction de* contraintes *spatiales* sur des espaces à placer et à dimensionner, en définissant la notion de *SCSP « Spatial Constraint Satisfaction Problem »*. La résolution d'un *SCSP* consiste à trouver les emplacements, les orientations et les dimensions des espaces pour satisfaire toutes les contraintes. Le Tableau 3.1. illustre l'analogie entre le formalisme d'un *CSP* et d'un *SCSP*.

CSP	*SCSP*
X : variable	O : pièce
D : domaine	G : configuration
C : contraintes	C : contrainte

Tableau 3.1 : Équivalence entre le formalisme d'un *CSP* et d'un *SCSP* selon [Charman, 1995]

Charman a défini une nouvelle cohèrence, appelé « *cohérence semi-géométrique* », qui est efficace pour la réduction, a priori, des configurations (domaine des variables géométriques des espaces) des espaces à placer c'est-à-dire qu'elle fonctionne en terme de réduction de domaine du couple (*x1, y1*) plutôt qu'en terme de réduction des domaines de *x1* et *y1* séparément. Pour mieux comprendre les mécanismes de cette nouvelle cohérence, nous avons illustré un exemple proposé par Charman : si deux espaces *o1* et *o2* sont à placer, et si *o1* est choisi en premier pour être placé, en instanciant les variables *x1, y1, dx1(longueur)* et *dy1(largeur)* de *o1*, des réduction de domaine de la configuration de *o2* seront automatiquement effectuées (cf. Figure 3.17).

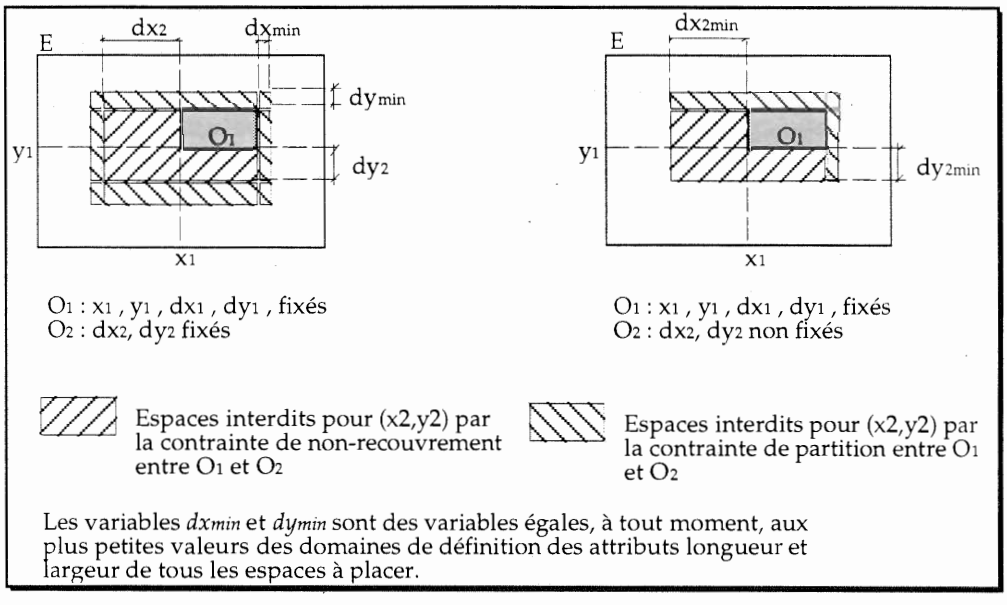


Figure 3.17 : Exemple de réduction de domaine [Charman, 1995] avec la cohérence semi-géométrique. On remarque les petits interstices laissés en blanc ; ils correspondent à des solutions adjacentes.

Aggoun et *Beldiceanu* [Aggoun et Beldiceanu, 1992] ont introduit la notion de *contrainte cumulative* dans le langage de programmation logique avec contraintes[28] *CHIP (Constraint Handling in Prolog)*. Cette contrainte a été développée pour des problèmes

28. Dans la littérature on trouve CLP : *Constraint Logic Programming* dont un autre exemple est Prolog III et depuis peu Prolog IV.

de planification et d'ordonnancement mais peut être utilisée très efficacement pour des problèmes de placement en architecture. Elle permet une réduction très efficace des domaines du problème étudié.

Définition 3.1 : D'un point de vue géométrique, la contrainte cumulative en *2D* exprime le fait que pour toute droite *D* verticale ou horizontale coupant l'*espace de placement*, la somme des dimensions des rectangles que coupe la droite est égale ou inférieure à la dimension de l'*espace de placement* (égale dans le cas de recouvrement total des espaces).

La contrainte *cumulative* procède de la manière suivante : soit N la longueur et la largeur de. l'*espace de placement*, n le nombre des espaces à placer, $x_i(i=1,\ldots,n)$ l'abscisse du point de référence de l'espace i, et $y_i(i=1,\ldots,n)$ l'ordonnée du point de référence de l'espace i, li $(i= 1,\ldots,n)$ est la longueur de l'espace i, $w_i(i=1,\ldots,n)$ est la largeur de l'espace i (cf. Figure 3.18).

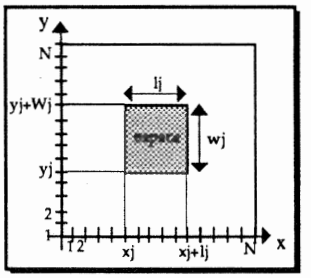


Figure 3.18 : Représentation géométrique d'un espace selon Aggoun et Beldiceanu

Les deux contraintes cumulatives qui, dans le cas du placement, s'appliquent dans les deux directions x et y (extension au 2D) , sont satisfaites si et seulement si :

$$\begin{cases} \forall i \in [1,N] \quad \sum_{j/x_j \le x_i \le +x_j+l_j} l_j \le N \\ \forall i \in [1,N] \quad \sum_{j/y_j \le y_i \le +y_j+w_j} w_j \le N \end{cases}$$

Dans les travaux déjà cités, ceux de *Charman*, d'*Aggoun* et *Beldiceanu* sont les plus pertinents ; ils proposent des contraintes très efficaces pour la réduction de l'espace de recherche.

Sur le plan de la classification des contraintes, *Charman* a présenté deux types de contraintes. Les contraintes *spécifiques* dépendent essentiellement du domaine d'application, ce sont des contraintes *fonctionnelles*. Les contraintes *implicites* sont prises en compte par défaut par le système. Les contraintes de *détection des incohérences*, de *symétrie* et celles *sur les rectangles* permettent de réduire considérablement la combinatoire en éliminant d'emblée les valeurs incohérentes des domaines des variables qui définissent le problème. Programmer une cohérence

comme la cohérence semi géométrique de *Charman* ou la contrainte cumulative de *Aggoun* est typiquement un travail de recherche en informatique. Nous n'avons pas l'ambition que ce travail le soit, et n'ayant pas développé sous *CHIP* et n'ayant pas pu disposer à temps de l'environnement de Charman[29], nous avons proposé des contraintes essentiellement de type *démons* en faisant intervenir des points de choix. Ainsi, au lieu de réduire les domaines de définition des variables a priori lors de la prise en compte de cette contrainte, une contrainte démon ne déclenche une propagation qu'a posteriori lors de l'instanciation d'une variable.

En revanche, bien que plus simples et en théorie efficaces, nos contraintes sont pour la plupart originales et ont l'avantage de se programmer plus simplement si le logiciel devait être réécrit ou commercialisé. Enfin, et ceci est très certainement l'argument le plus important, nous montrons clairement au *chapitre 4* que l'efficacité des contraintes devient un critère d'influence négligeable au regard de l'amélioration qu'apporte notre algorithme d'énumération de solutions topologiques (cf. *chapitre 4*).

3.2.2. Notre modèle de contraintes

A partir de nos hypothèses de départ et des contraintes déjà proposées, nous avons défini trois groupes de contraintes :

- (1) **les contraintes fonctionnelles** : elle regroupent les contraintes qui permettent de décrire le cahier des charges d'un bâtiment,
- (2) **les contraintes implicites** : ce sont des contraintes géométriques implicitement prises en compte dans tous les problèmes de placement,
- (3) **les contraintes de réduction de l'espace de recherche** : elles regroupent la contrainte d'élimination des espaces incohérents [Charman, 1995], la contrainte de symétrie, la contrainte de réduction des topologies et la contrainte de déduction des orientations.

Les contraintes proposées dans ARCHiPLAN sont linéaires et/ou non-linéaires. Ce sont essentiellement des égalités ou des inégalités.

3.2.2.1. Les contraintes fonctionnelles

C'est à partir de ces contraintes que l'architecte pourra d'une manière déclarative décrire le cahier des charges d'un bâtiment. Les contraintes fonctionnelles regroupent :

29. Nous avons bénéficié d'un accord de prêt gratuit établi avec l'INRIA pour une version d'EAAS [Charman, 1995]. Cela nous a permis de faire des tests comparatifs d'EAAS et d'ARCHiPLAN.

- les contraintes unaires,
- les contraintes d'adjacence entre les locaux,
- les contraintes d'adjacence avec l'*espace* de *placement*,
- les contraintes d'adjacence avec les escaliers,
- les contraintes d'orientation relatives,
- les contraintes de ratio,
- les contraintes de communication entre les étages,
- la composition de contraintes fonctionnelles.

• Les contraintes unaires

Les contraintes unaires sont posées sur les attributs d'un seul espace. Ces contraintes permettent d'imposer une valeur minimale ou maximale aux attributs géométriques. Les différentes contraintes unaires sont :

- *Domaine-x1* (IN : val-min, val-max, espace) : cette contrainte impose un domaine à l'abscisse *x1* du premier point de référence de l'espace sur lequel elle est posée.

La même contrainte se retrouve pour *y1*, *x2*, *y2*, *S*, *L*, *W*.

- *Impose-Orientation* (IN : orientation, espace) : cette contrainte impose une seule orientation à *espace* afin que *L* et *W* correspondent à l'horizontale ou à la verticale.

• Les contraintes d'adjacence

Les contraintes d'adjacence sont fondamentales pour la description d'un cahier des charges, le fait de concevoir étant en grande partie le fait de fixer l'adjacence des pièces et des circulations entre elles (cf. Figure 3.19) ou de fixer la distance entre deux pièces. la notion d'adjacence peut aller jusqu'à la spécification des cotés des espaces qui doivent être adjacents. En réalité, nous avons développé des contraintes d'*adjacence généralisées*, c'est-à-dire pas forcément réduites au contact direct, mais permettant de maîtriser une distance entre deux pièces.

Dans un premier temps et avant de présenter la contrainte d'adjacence généralisée, nous détaillerons les quatre contraintes d'adjacence d'une pièce au côté spécifique d'une autre pièce (*Adjacent-au-nord*, *Adjacent-au-sud*, *Adjacent-a-ouest*, *Adjacent-a-est*). Pour ce faire, nous introduisons deux notions : celle de *longueur de contact* et celle de

périmètre de *protection*.

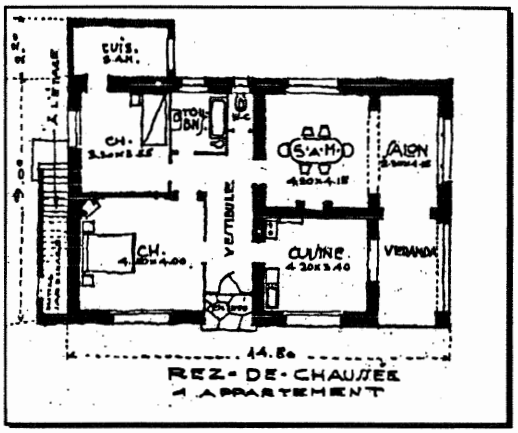


Figure 3. 19 : Exemple de pièces adjacentes dans un appartement

- **Notion de *Longueur de contact* (*d1*)** : La *longueur de contact d1* (cf. Figure 3.20) est une variable contrainte entière exprimant la longueur de communication entre deux espaces. Cette variable est présente dans toute contrainte d'adjacence. Par défaut, $Min(D(d1))=0$ et $Max(D(d1))=+\infty$. En pratique, elle est bien sûr utilisée pour imposer une communication de largeur minimale pour le passage des personnes : $Min(D(d1))=d1min>0$.

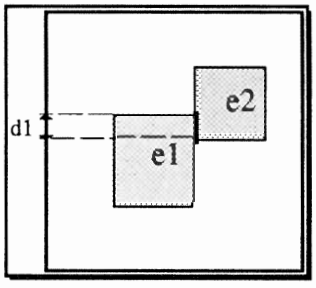


Figure 3.20 : *Longueur de contact* (*d1*) entre deux espaces *e1* et *e2*

- **Notion de *périmètre de protection* d'un espace** : la notion d'adjacence directe (contact entre deux espaces) a été généralisée à la maîtrise de la distance entre deux espaces (cf. Figure 3.21). Cette distance *d2* est également une variable contrainte entière. Par défaut, son domaine est réduit à l'unique valeur 0, on retrouve alors la notion classique d'adjacence. Souvent, il est nécessaire pour isoler certains locaux techniques (stockage de produits dangereux...) d'imposer un *périmètre de protection* ; cela correspond à $Min(D(d2))=d2min>0$ et $Max(D(d2))=+\infty$. En toute généralité, on peut également imposer que deux espaces ne soient pas distants de plus d'une certaine valeur, $Max(D(d2))=d2max>0$ et $Min(D(d2))=0$, ou encore que deux espaces soient éloignés d'une certaine distance, ni trop petite, ni trop grande, ce qui se traduit par : $Min(D(d2))=d2min>0$ et $Max(D(d2))=d2max$.

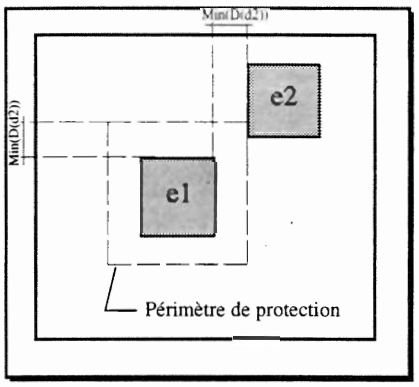


Figure 3. 21 : Périmètre de protection (d2) de l'espace e1

Il est maintenant possible de décrire les 4 contraintes d'adjacence généralisée spécifiques à une orientation donnée (est, ouest, nord, sud). Par la suite, nous n'en détaillons qu'une, les trois autres sont données en *annexe A.4.1.*

- **La contrainte *Adjacent-au-nord***

Comme il est indiqué dans la Figure 3.22, cette contrainte impose que l'espace *e1* soit au nord de l'espace *e2*. Une *longueur de contact d1* et un *périmètre de protection d2* peuvent être pris en compte. Nous retrouvons l'adjacence au nord classique lorsque : *Min(D(d1))=0* et *Max(D(d2))=0*. On peut remarquer qu'on peut faire porter l'*adjacence au nord* sur un coté particulier (*L* ou *W*) de *e1* ou *e2* en instanciant dès le départ un attribut d'orientation à 0° ou 90°.

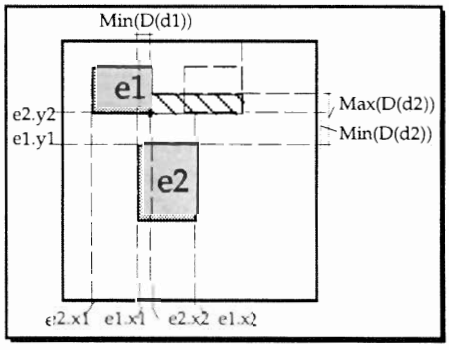


Figure 3.22 : Le rectangle hachuré indique les positions permises du point de référence (*x2, y2*) de l'espace *e1* par la contrainte *e1 Adjacent-au-nord e2*

La contrainte d'adjacence au nord est indiquée dans la Figure 3.23.

Contrainte **Adjacent-au-nord** (IN : e1, e2, d1, d2)

→ e1.y2=e2.y1-d2
e1.x2≥e2.x1+d1
e1.x1≤e2.x2-d1

fin Contrainte

Figure 3.23 : La contrainte *Adjacent-au-nord*

• **la contrainte *Adjacent***

Elle fait intervenir une nouvelle variable contrainte dite variable d'*adjacence* à valeur dans l'ensemble discret {*E,O,N,S*}. Cette contrainte *adjacent* est une contrainte "démon" qui à chaque instanciation de la variable d'*adjacence* déclenche une méthode de propagation. Par contre, pour éviter les solutions redondantes au *nord-est, nord-ouest, sud-est* et *sud-ouest,* on décide pour les choix E et O d'éliminer les solutions correspondant à nord-est, sud-est, nord-ouest et sud-ouest. On réalise ainsi un partitionnement de l'orientation relative, que l'on retrouvera et que l'on détaillera pour la contrainte de *non-recouvrement*. Les méthodes de propagations déclenchées correspondant à chaque choix sont indiquées dans la Figure 3.24.

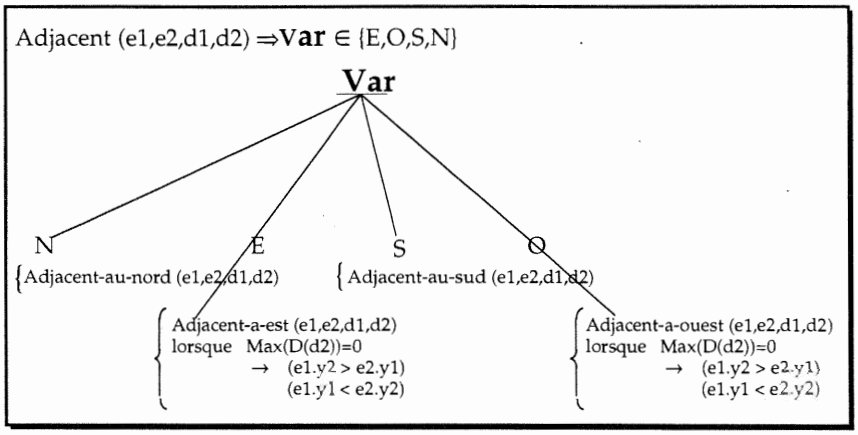


Figure 3.24 : Les méthodes de propagation de la contrainte d'adjacence

• **le cas particulier de la contrainte d'adjacence entre les couloirs**

Il peut apparaître dans le cas particulier de l'adjacence entre deux couloirs des solutions qui sont fonctionnellement identiques (cf. Figure 3.25). La composition des deux couloirs ne forme en réalité qu'un seul couloir. Ce qui n'est pas le cas pour les pièces.

Figure 3.25 : Exemple de solutions fonctionnellement identiques

Afin d'éliminer ces solutions dites *fonctionnellement identiques,* nous avons proposé une contrainte dont le principe est le suivant : à chaque fois que deux couloirs c1 et c2 de même largeur sont dans le prolongement l'un de l'autre, la longueur de c1 est toujours égale à la valeur minimale de son domaine tant que la longueur de c2 est inférieure à la valeur maximale de son propre domaine (cf. Figure 3.26).

Contrainte	**élimine-redondance-couloir** (IN : c1, c2)
tant que	V(D(c2.L)<Max(D(c2.L)) → V(D(c1.L)) = Min(D(c1.L))
fin Contrainte	

Figure 3.26 : Contrainte d'élimination des solutions dites *fonctionnellement identiques*

- **Les contraintes d'adjacence avec l'espace de placement**

Cette contrainte permet d'imposer à un espace d'être contre l'une des façades de l'*espace de placement* (pour bénéficier de l'éclairage naturel par exemple). Dans un premier temps et avant de présenter la contrainte d'adjacence généralisée avec l'*espace de placement*, nous détaillerons les quatre contraintes d'adjacence d'une pièce avec un côté spécifique de l'*espace de placement* (*Sur-la-façade-nord, Sur-la-façade-est, Sur-la-façade-sud, Sur-la-façade-ouest).*

Il est maintenant possible de décrire les 4 contraintes d'adjacence avec l'*espace de placement*. Nous nous limitons à présent à la contrainte *Sur-la-façade-nord ; on* se reportera à l'*annexe A.4.2.* pour les trois autres.

- **la contrainte *Sur-la-façade-nord***

Cette contrainte impose que l'espace *e1* soit toujours adjacent au côté nord de l'*espace de placement E* (cf. Figure 3.27).

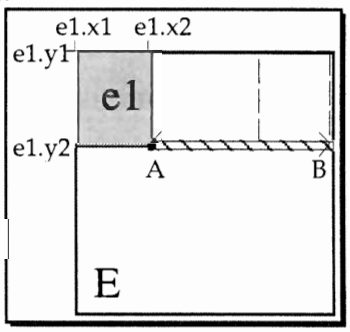


Figure 3. 27 : Le segment [AB] indique les positions permises du point (*x2, y2*) de l'espace *e1* par la contrainte *Sur-la-façade-nord*

Contrainte	**Sur-la-façade-nord** (IN : e1, E)
→	**e1.y1=E.y1** **e1.x1≥E.x1** **e1.x2≤E.x2**
fin Contrainte	

Figure 3. 28 : La contrainte *Sur-la-façade-nord*

• **la contrainte *Sur-le-contour***

Elle est représentée par une variable contrainte à quatre valeurs {*E,O,S,N*}. A chaque instanciation de cette variable une propagation est déclenchée qui correspond globalement à une contrainte d'adjacence de base avec l'*espace de placement*. On évite simplement d'énumérer deux fois les quatre positions correspondant aux coins de l'*espace de placement* (cf. Figure 3.29), ce qui correspond au partitionnement déjà évoqué pour la contrainte d'adjacence. Les méthodes de propagation déclenchées correspondent au quatre choix qui sont indiqués dans la Figure 3.29.

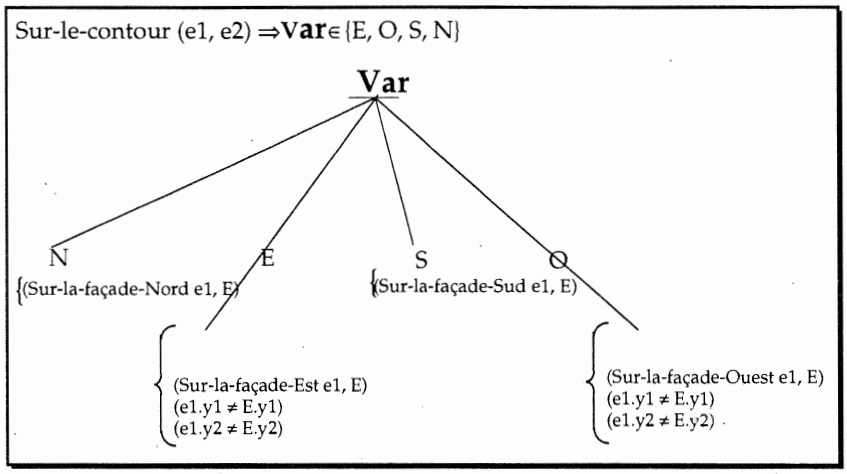


Figure 3.29 : Les méthodes de propagation de la contrainte *Sur-le-contour*

• **Les contraintes d'adjacence avec les escaliers**

Deux contraintes spécifiques à l'escalier ont été définies, celle qui impose à un espace d'être toujours adjacent à la première marche pour assurer l'accès à l'escalier et celle qui impose à un espace d'être toujours adjacent avec la marche d'arrivée pour assurer l'accès à l'étage d'arrivée.

• **la contrainte *Adjacent-avec-la-marche-de-départ***

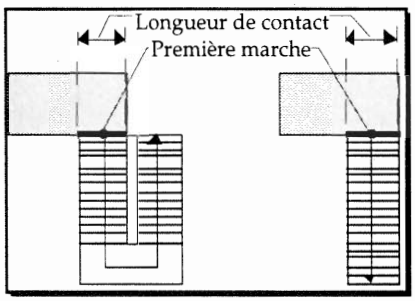


Figure 3.30 : Adjacence de la marche de départ d'un escalier avec un couloir

Contrainte **Adjacent-avec-marche-de-départ** (IN : escalier, pièce)

lorsque escalier.orientation≡0°

→ escalier.y.marche.de.depart=pièce.y2
x2.pièce ≥ escalier.x-marche-de-depart + (escalier.L-marche/2)
x1.pièce ≤ escalier.x-marche-de-depart - (escalier.L-marche)

lorsque escalier.orientation≡90°

→ escalier.x.marche.de.depart=pièce.x1
y2.pièce ≥ escalier.y-marche-de-depart + (escalier.L-marche/2)
y1.pièce ≤ scalier.y-marche-de-depart - (escalier.L-marche)

lorsque escalier.orientation≡180°

→ escalier.y.marche.de.depart=pièce.y1
x2.pièce ≥ escalier.x-marche-de-depart + (escalier.L-marche/2)
x1.pièce ≤ escalier.x-marche-de-depart - (escalier.L-marche)

lorsque escalier.orientation≡270°

→ escalier.x.marche.de.depart = pièce.x2
y2.pièce ≥ escalier.y-marche-de-depart + (escalier.L-marche/2)
y1.pièce ≤ escalier.y-marche-de-depart - (escalier.L-marche)

fin Contrainte

Figure 3.31 : Contrainte d'adjacence avec l'escalier pour la montée

- **Les contraintes d'orientation relative**

Une contrainte d'*orientation relative* impose à un espace *e1* relativement àun espace *e2* d'être, respectivement au nord, à l'est, au sud et à l'ouest. Il est indiqué dans les Figures 3.32 et 3.33 la contrainte d'orientation relative *au-nord-de* entre deux espaces.La partie hachurée indique les positions permises pour le point *(x2,y2)* de l'espace *e1* avec *e2* fixe. Les trois autres contraintes d'orientation relatives (*Au-sud-de, A-est-de, A-ouest-de*) sont décrites en *annexe 4.3.* cette contrainte est équivalente à Adjacent (e1,e2, d1=[0,+∞],d2=[0,+∞]) mais elle permet de faire l'économie de deux variables contraintes *d1* et *d2* et permet donc de ne pas avoir à les énumérer.

- **la Contrainte *Au-nord-de***

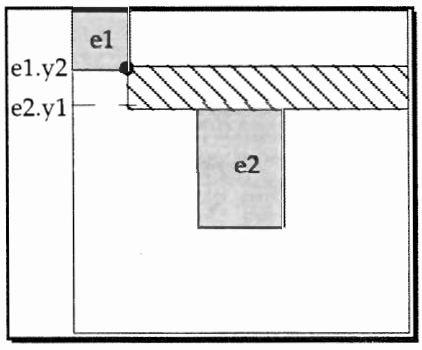


Figure 3.32 : Positions permises pour le point *(x2,y2)* de l'espace *e1* par la contrainte *Au-nord-de*

```
Contrainte      Au-nord-de (IN : e1, e2)
→ e1.y2≤e2.y1
fin Contrainte
```

Figure 3. 33 : La contrainte *Au-nord-de*

- **La contraintes de ratio entre les espaces**

Nous avons développé une contrainte de *ratio* entre deux paramètres géométriques *p1* et *p2* de deux espaces. Cette contrainte est très importante car elle permet d'imposer des proportions dans des limites acceptables, ce qu'un architecte désirera souvent faire. Notre première démarche a été de vouloir modéliser un *ratio* sous la forme d'une variable contrainte réelle. Comme nous utilisons la cohérence par arcs sur les entiers, c'était impossible. Nous modélisons les deux bornes du ratio *R* par les numérateurs et les dénominateurs de deux rationnels : $a1/b1 \leq R=(P1/P2) \leq (a2/b2)$. La contrainte *ratio-entre-deux-paramètres* est donnée en Figure 3.34

```
Contrainte      ratio-entre-deux-paramètres (IN : p1, p2,a1,b1,a2,b2)

→       a1.p2 ≤ b1.p1
        a2.p2 ≥ b2.p1

fin Contrainte
```

Figure 3.34 : La contrainte de *ratio* entre deux paramètres contraints

On remarque tout d'abord que, *p1*, *p2* et les bornes de *R* sont positifs pour ne pas avoir d'ambiguïté avec les inégalités (au niveau de la multiplication en croix). Cette contrainte de *ratio* sera donc utilisée pour exprimer des ratios entre des distances comme : *L*, *W*, *S*, *d1*, *d2*.

Exemple : on veut que la surface du WC soit comprise entre 0,4 et 0,5 fois celle de la SDB, on écrit : ratio-entre-deux-paramètres (wc.S, SDB.S, 2, 5, 1, 2). Cette contrainte peut également exprimer la *règle d'or de Le Corbusier*.

- **Les contraintes de communication entre les étages**

Ces contraintes nous ont permis de généraliser notre application à des bâtiments à plusieurs étages.

• **la Contrainte *Sur***

Cette contrainte impose à l'étage supérieur d'avoir mêmes longueur et largeur que l'étage inférieur, seule l'altitude est différente. Étant donné que nous travaillons en deux dimensions, nous ne tenons pas compte de l'altitude Z.

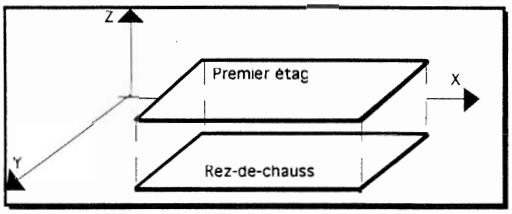


Figure 3.35 : La contrainte Sur

• **la Contrainte *Communique-entre***

Cette contrainte a trois arguments : l'étage supérieur, l'étage inférieur et les circulations verticales. Si la circulation est un escalier (cf. Figure 3.36), ces dernières vont permettre la communication entre les deux étages. Chaque escalier se retrouvera au niveau des deux étages avec les mêmes attributs.

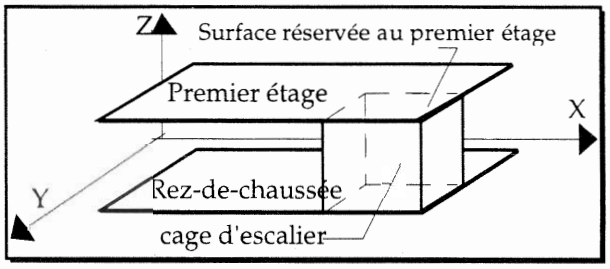


Figure 3. 36 : La contrainte *Communique-entre*

• **La disjonction de contraintes fonctionnelles**

la disjonction consiste à avoir un choix entre plusieurs alternatives ; par exemple, un espace *e1* est *Adjacent-a-est* ou *Adjacent-a-ouest* de *e2*. ARCHiPLAN permet actuellement de représenter la forme disjonctive suivante :

(*e1* (*conjonction_contraintes*) *e2*) *ou* (*e1* (*conjonction_contraintes*) *e3*) ou e1, e2 et e3 sont trois espaces et il s'agit d'un *ou* inclusif.

Cette disjonction engendre toujours *un point de choix*. Dans le cas où cette contrainte met en jeu trois espaces, il apparaît des solutions redondantes. Si, par exemple, un espace *e1* est adjacent à un espace *e2* « *ou* » adjacent à un espace *e3*, les solutions où l'espace *e1* est adjacent aux deux espaces *e2* et *e3* risquent d'apparaître pour chaque

choix. Nous avons mis au point une contrainte nous permettant d'éliminer ces redondances ; le principe en est l'énumération de toutes les solutions du premier choix, par contre de ne pas énumérer pour le deuxième choix les solutions respectant le premier.

Nous n'avons pas généralisé cette contrainte de disjonction à des formules propositionnelles complètes (combinaisons de ET, OU, et NON) à cause de la complexité du problème d'élimination des solutions redondantes. Nous rappelons que cette non-redondance est une contrainte que nous avons constamment à l'esprit pour que nos futures solutions topologiques soient distinctes (cf. *chapitre 4* cette forme particulière de disjonction a été choisie car c'est celle qui apparaît pour des adjacences avec une circulation en deux parties (en *L* ou en *T*), une pièce étant alors adjacente à au moins l'une des deux parties.

3.2.2.2. Les contraintes implicites

Les contraintes implicites sont déclenchées par défaut par ARCHiPLAN. Dans des cas très particuliers, l'utilisateur aura loisir de désactiver l'une de ces contraintes. Ces contraintes, somme toute classiques, regroupent :

- la contrainte de *non-recouvrement*,
- la contrainte d'*inclusion*,
- la contrainte de *recouvrement total de l'espace de placement*.

• La contrainte de non-recouvrement

La contrainte de *non-recouvrement* exprime le fait que les espaces ne peuvent pas se chevaucher deux à deux. Cette contrainte, qui s'applique donc entre tout couple d'espaces fait apparaître une nouvelle variable d'*orientation relative* à 4 valeurs {*E*,*O*,*S*,*N*} pour chaque couple. Soit n le nombre d'espaces on aura donc $\left[\frac{1}{2}n(n-1)\right]$ variables d'*orientation relative*. Cette nouvelle variable partitionne le voisinage d'un espace en quatre (cf. Figure 3.37). Sur cet exemple, on constate que les valeurs *N* et *S* adressent plus de solutions que les valeurs *E* et *O*. Ce partitionnement dissymétrique nous permet d'éviter deux fois les mêmes solutions (au nord-est, nord-ouest, sud-est et sud-ouest). Éviter ces solutions identiques nous garantit l'impossibilité d'avoir pour deux solutions topologiques différentes la même solution numérique. Nous verrons que c'est l'instanciation de ces $\left[\frac{1}{2}n(n-1)\right]$ variables qui, si elle s'avère cohérente, donne une solution topologique (cf. *chapitre 4*).

Nous illustrons dans la Figure 3.37 les positions permises pour $e2.x2$ et $e2.y2$ par la contrainte de *non-recouvrement* entre les espaces $e1$ et $e2$. Cette contrainte de *non-recouvrement* dépend de la notion de *longueur* et de *largeur minimale*. Cette longueur minimale *Lmin* et cette largeur minimale *Wmin* sont des variables égales, à tout moment, aux plus petites valeurs des domaines de définition des attributs *longueur* et *largeur* de tous les espaces à placer. Ces valeurs sont utiles pour contraindre deux espaces à, être adjacents, ou bien suffisamment distants pour que puisse s'intercaler un autre espace ; ce minimum s'écrit *dmin=Min(Lmin,Wmin)*. Nous n'avons pas tenu compte de la notion de *longueur* et de *largeur minimale* pour les deux zônes *N* et *S*. Cette prise en compte aurait suscité la partition de la zone *N* et *S* en trois sous-zônes, puisque *Lmin* et *Wmin* ne sont pas à éliminer sur toute la longueur des zones *N* et *S* (cf. la Figure 3.37). Cette option fait intervenir trop de points de choix supplémentaires.

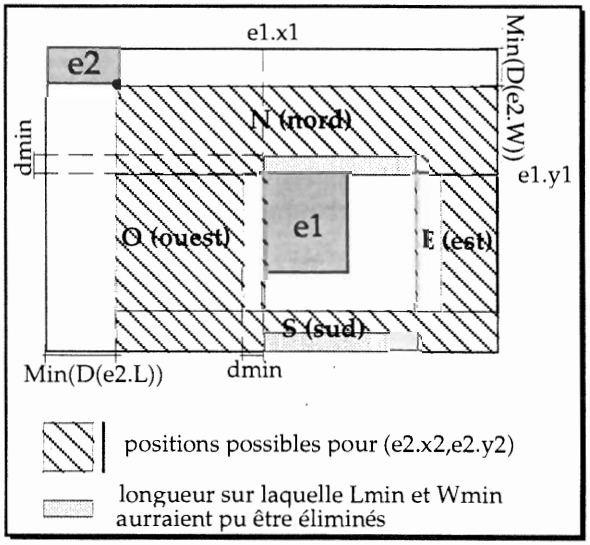


Figure 3.37 : Positions permises pour le point $(x2,y2)$ de l'espace $e2$ par une contrainte de *non-recouvrement* avec l'espace $e1$. La partition du voisinage d'un espace en $\{E,O,N,S\}$ est donné.

A chaque instanciation d'une variable d'*orientation relative*, une méthode de propagation est déclenchée, qui réduit le domaine des différents attributs géométriques des espaces deux à deux ; ces méthodes sont indiquées en Figure 3.38.

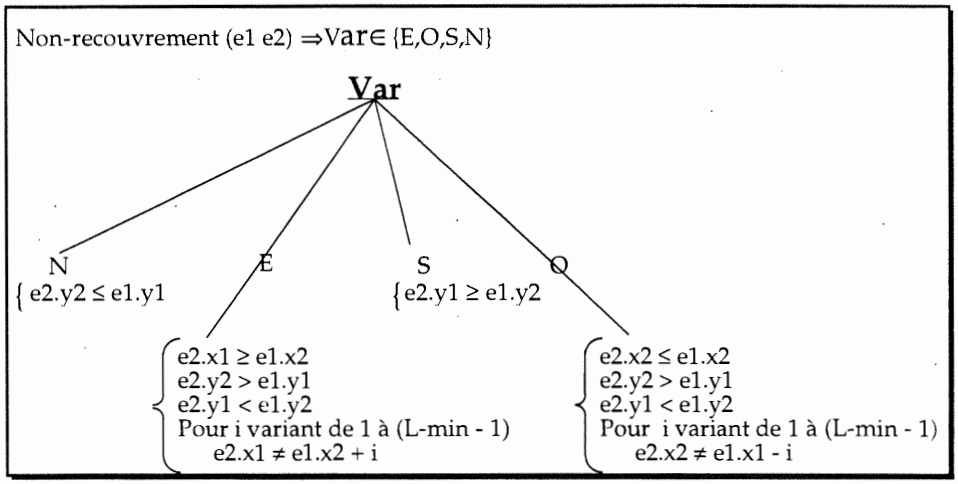


Figure 3.38 : Les méthodes de propagation d'une contrainte de *non-recouvrement*

En fait, nous posons une contrainte de *non-recouvrement* entre deux espaces *e1* et *e2* lorsqu'il n'existe pas de contrainte d'adjacence ou d'orientation relative entre eux déjà présente dans le graphe fonctionnel. Nous avons l'équivalence suivante :

Non-recouvrement (e1,e2) ≡ *Adjacent* (e1 e2 d1 d2) avec d1∈ [0 +∞] et d2∈ [0 +∞].

- **La contrainte d'inclusion**

La contrainte d'*inclusion* impose à chaque espace d'être dans l'*espace de placement*.

Contrainte Dans **(IN : e1, E)**

→ e1.x1 ≥ E.x1
e1.x2 ≤ E.x2
e1.y1 ≥ E.y1
e1.y2 ≤ E.y2

fin Contrainte

Figure 3.39 : Contrainte d'inclusion

- **La contrainte de recouvrement total de l'espace de placement**

La contrainte de *recouvrement total de l'espace de placement* exprime le fait qu'on ne désire généralement pas, en architecture, d'espace perdu et donc, que la somme des surfaces des *n* espaces est égale à la surface de l'*espace de placement*. Pour des cas particuliers, cette contrainte pourra être désactivée.

$\sum_{i=1}^{i=n}$ Si = Surface de l'*espace de placement*

3.2.2.3. Les contraintes de réduction de l'espace de recherche

Ces contraintes qui permettent de réduire énormément la combinatoire, sont spécifiques à notre approche. Elles regroupent :

- la contrainte d'*élimination des espaces incohérents,*
- la contrainte de *symétrie,*
- la contrainte de *réduction des topologies,*
- la contrainte de *transitivité des orientations.*

• La contrainte d'élimination des espaces incohérents

Cette contrainte, déjà proposée par Charman [Charman, 1995], dépend également des notions de *longueur* et de *largeur minimales.* Ces valeurs sont déterminées pour contraindre chaque espace à, être adjacent, ou bien être distants d'une valeur suffisante pour que puisse s'intercaler un espace entre lui et le contour de l'*espace de placement* ; cette "valeur suffisante" est la distance *dmin=Min(Lmin,Wmin).* Cette contrainte va de paire avec la contrainte de *recouvrement total de l'espace de placement.* La Figure 3.40 illustre les positions permises pour le point (*x1,y1*) de l'espace *e1* par la contrainte d'*élimination des espaces incohérents.*

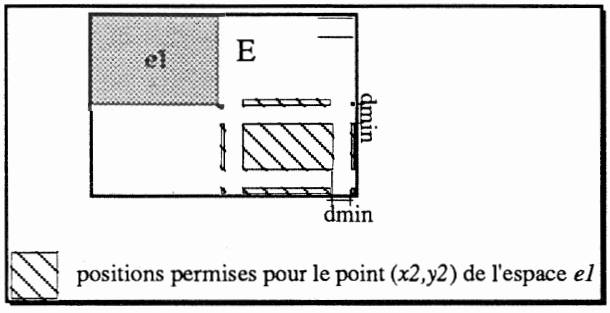


Figure 3.40 : Positions permises pour le point (*x2,y2*) de l'espace *e1* par la contrainte d'*élimination des espaces incohérents*

Contrainte **Elimine-incohérence** (IN : e1, E)

Pour i variant de 1 à (dmin - 1)
→ e1.x1 ≠ E.x1+i
e1.x1 ≠ E.x2 - e1.L + i

Pour j variant de 1 à (dmin - 1)
→ e1.y1 ≠ E.y1+j
e1.y1 ≠ E.y2 - e1.W + j

fin Contrainte

Figure 3.41 : Description de la contrainte *élimination des espaces incohérents.*

• Les contraintes de symétrie

Les contraintes de *symétrie* ont pour but d'éviter les combinaisons des solutions portant sur des espaces de même type soumis exactement aux mêmes contraintes, c'est-à-dire aux mêmes domaines de définition initiaux et aux mêmes contraintes avec de tierces pièces. Par exemple, dans un F5, il peut y avoir 3 chambres ayant les mêmes contraintes initiales de dimensionnement et les mêmes contraintes d'adjacence avec la circulation.

Afin d'éliminer les combinaisons symétriques entre deux espaces *e1* et *e2*, il suffit d'imposer que *e1.x1* soit toujours inférieur ou égal à *e2.x1* et lorsque *e1.x1=e2.x1*, il faut imposer que *e1.y1<e2.y1* (cf. Figure 3.42). Cette procédure est appliqué pour *n* espaces symétriques dans la Figure 3.45.

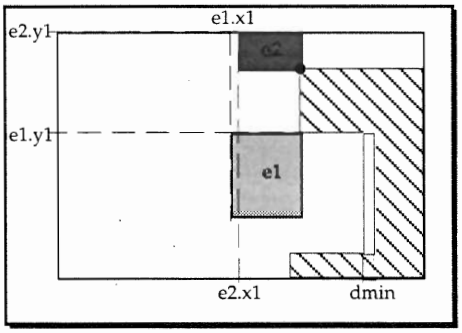


Figure 3.42 : Positions permises pour le point (*x2,y2*) de l'espace *e2* par les contraintes de *symétrie* et de *non-recouvrement*

Contrainte **Elimine-symétrie** (*l* : Liste-espaces-symétriques)
n = longueur (*l*)
pour i variant de 1 à *n*
ei = element (i, *l*)
pour j variant de (i + 1) à *n*
eJ = element (j, *l*)
→ $e_i.x1 \leq e_j.x1$
lorsque $V(e_i.x1) \equiv V(e_j.x1)$
→ $e_i.y1 < e_j.y1$
fin Contrainte

Figure 3.43 : contrainte de *symétrie*

• généralisation de la contrainte de symétrie aux différentes orientations

La contrainte de symétrie doit prendre en compte les différentes orientations des espaces. Dans le cas où deux espaces *e1* et *e2* symétriques ont deux orientations

possibles, la contrainte de symétrie définie précédemment sera propagée à chaque fois que les instances des orientations des deux espaces seront égales. Dans le cas contraire, lorsque les orientations sont différentes (*V(D(orientation.e1))*=0° et *V(D(orientation.e2))*=90°), il n'existe pas de solution symétrique. Par contre, toutes les solutions correspondant à *V(D(orientation.e1))*=90° et *V(D(orientation.e2))*=0° sont équivalentes à celles correspondant à *V(D(orientation.e1))*=0° et *V(D(orientation.e2))*=90°. Pour éliminer ces redondances, il suffit d'énumérer une seule fois les cas où les deux orientations sont différentes. La Figure 3.44 illustre la contrainte de symétrie généralisée à plusieurs orientations.

Contrainte **Symétrie** (*l* : Liste-espaces-symétriques)
n = longueur (*l*)
pour i variant de 1 à *n*
    ei = element (i, *l*)
    pour j variant de (i + 1) à *n*
        lorsque V(ei.orientation)≡V(ej.orientation)
        (**Elimine-symétrie** («ei, ej»))
        lorsque V(ei.orientation)≠V(ej.orientation)
        lorsque v(ei.orientation)≡90°
        v(ej.orientation)≠0°
fin Contrainte

Figure 3. 44 : La contrainte de *symétrie* généralisée aux différentes orientations

- **La contrainte de réduction des topologies**

La contrainte de *réduction des topologies* est opérationnelle lorsqu'il existent dans le cahier des charges des contraintes d'adjacence avec l'*espace de placement*. La contrainte de réduction opère de la manière suivante : si un espace est *Sur-la-façade-nord* de l'*espace de placement*, aucun espace ne peut être au nord de ce dernier. La valeur *N* est alors retirée du domaine de définition de toutes les variables d'*orientation relative* émanant des contraintes de *non-recouvrement* (précédemment évoquées) avec cet espace. En réduisant le domaine de définition de ces variables, on élimine du même coup une partie des topologies incohérentes.

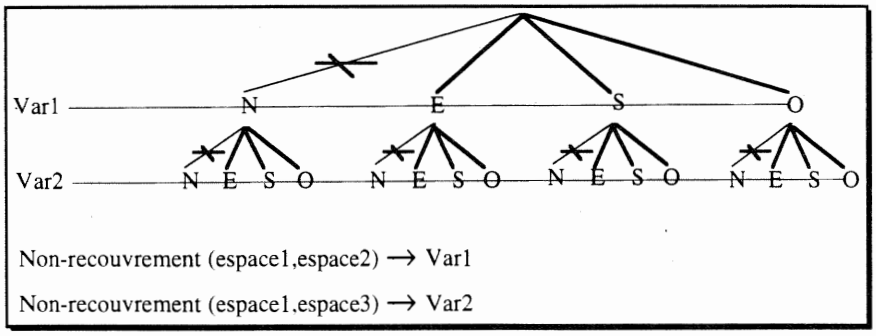


Figure 3.45 : Élimination des choix *N* (« au nord de ») du domaine des variables d'orientation relative

La contrainte de réduction des topologies est généralisée aux autres orientations :

- Si un espace est *Sur-la-façade-sud* aucun espace ne peut être au sud de celui-ci,
- Si un espace est *Sur-la-façade-est* aucun espace ne peut être à l'est de celui-ci,
- Si un espace est *Sur-la-façade-ouest* aucun espace ne peut être à l'ouest de celui-ci.

- **La contrainte de transitivité des orientations**

La contrainte de *transitivité des orientations* traduit des règles géométriques élémentaires qui permettent de déduire, par transitivité, les orientations des espaces, c'est-à-dire d'instancier directement des variables d'*orientation relative* émanant des contraintes de *non-recouvrement*. Si, par exemple, un espace *e1* est au nord d'un espace *e2* et si *e2* est lui-même au nord d'un espace *e3* alors *e3* est au nord de *e1*. Nous avons développé la contrainte de transitivité pour les orientations relatives au nord (*N*) et au sud (*S*). Nous avons évité son utilisation pour les orientations relatives à l'est (*E*) et à l'ouest (*O*) parceque la partition {N, E, O, S} rend cette transitivité impossible dans ces deux cas.

3.3. Conclusion

La base de connaissances ARCHiPLAN est constituée d'un ensemble de classes et de contraintes. Nous avons tenu compte des aspects fondamentaux de notre démarche de départ :

- la modélisation des contraintes d'*adjacence* et de *non-recouvrement* comme une composition de quatre orientations de base nous donne la possibilité de concevoir un algorithme d'énumération des topologies. La différence topologique entre deux espaces réside ainsi dans les quatre orientations cardinales. Nous l'aborderons plus en détail au chapitre suivant.

- l'existence de la classe escalier et des contraintes de placement spécifiques à cette classe nous permet de résoudre des problèmes de bâtiments à plusieurs étages,
- nous avons proposé de nouvelles contraintes :
  - la contrainte de classe d'*élimination des redondances* des *pièces* pour les deux orientations,
  - les contraintes de classe de *maintien de cohérence fonctionnelle* des escaliers à chaque orientation,
  - la contrainte d'*adjacence généralisée* à la *longueur de contact* et au *périmètre de protection*,
  - la contrainte de *ratio* entre deux paramètres géométriques,
  - la contrainte de *symétrie généralisée* aux deux orientations des espaces,
  - la contrainte de *réduction des topologies*.

Chapitre 4

Les algorithmes de placement

Nous avons vu au *chapitre 3* les principales entités et contraintes qui permettent de décrire un cahier des charges. A ce stade, nous n'avions pas encore abordé la recherche de solutions, qu'il s'agisse d'esquisses (solutions géométriques « floues ») ou de solutions numériques (où chaque variable est instanciée). Nous allons présenter dans ce chapitre les principaux algorithmes qui permettent de passer du schéma fonctionnel (cf. *chapitre 2*) à l'énumération des esquisses puis à celle des meilleures solutions numériques de chaque esquisse au regard de critères à minimiser.

4.1. État de l'art sur les stratégies de résolution

De nombreux travaux ont déjà proposé des stratégies de résolution de problèmes de placement utilisant des techniques de propagation de contraintes ou des systèmes experts. Le domaine d'application est très large. Il concerne l'allocation spatiale en architecture [Charman, 1995 ; Maculet, 1991], l'aménagement spatial (aménagement de salles informatique, de cuisines...) [Baykan et Fox, 1991], l'urbanisme [Villamayor, 1980], l'architecture navale [André, 1986], l'architecture des microprocesseurs[30] [DuVerdier, 1993] et l'ingénierie spatiale (aménagement de panneaux de satellites) [Aerospatiale, 1992]. L'analyse de ces travaux a été effectuée à trois niveaux :

- (1) sur les stratégies générales de résolution,
- (2) sur les heuristiques introduites pour rendre plus efficace la recherche de solutions,
- (3) sur les types de contraintes utilisés (ils ont déjà été vus au *chapitre 3*.

4.1.1. Les stratégies générales de résolution

Différentes stratégies ont été proposées. Elles présentent des différences compte tenu des techniques utilisées et des algorithmes mis au point. Nous présentons par la suite les approches : *Generate and Test, planification de l'aménagement,* optimisation *mathématique, système expert* et *programmation par contrainte.*

4.1.1.1. L'approche Generate and Test

Fletcher [Fletcher, 1965] a appliqué la technique du *Generate and Test* qui consiste à générer l'ensemble des solutions numériques pour un élément à placer puis à vérifier à l'aide des tests leur compatibilité avec l'ensemble des contraintes posées au départ.

30. les VLSI : Very Large Scale Integration

Cette méthode est illusoire pour des problèmes de taille pratique car la combinatoire apparente[31] est trop importante. Nous verrons que les méthodes de programmation par contraintes résolvent en partie ce problème en réduisant *au plus tôt* les domaines des différentes variables du problème à la pose des contraintes d'une part (propagation statique) et au cours de l'instanciation (propagation dynamique).

4.1.1.2. L'approche dite de "planification de l'aménagement"

Parmis les travaux qui ont utilisé cette approche nous avons ceux de Eastman [Eastman, 1973] et de Pfefferkorn [Pfefferkorn, 1975].

- Dans GSP, « *General Space Planner* » qu'a développé *Eastman*, les objets à placer sont de forme rectangulaire et de dimensions fixées. Au cours de l'énumération des solutions, à chaque fois qu'un objet est placé, par projection des cotés de l'objet sur ceux de l'espace de placement, l'algorithme définit des points de position qui seront utilisés pour le placement des autres objets. La Figure 4.1 montre l'apparition de points de position après avoir placé le premier objet. Il en résulte 24 points. Ainsi, 96 configurations, correspondant aux quatre orientations possibles par objet, seront testées pour le placement du prochain objet. Le système de génération des points de position ne permet pas d'obtenir toutes les solutions. L'impossibilité de définir des domaines de valeurs aux variables restreint les possibilités d'utilisation. GSP reste l'un des premiers systèmes d'aménagement spatial développé en 1973 dans le cadre de la recherche.

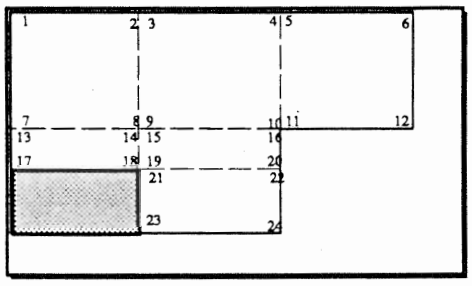


Figure 4.1 : Discrétisation dynamique de l'espace de placement (24 points de position) après le placement de chaque objet [Eastman, 1973]

- *Pfefferkorn* a développé le système DPS « *Designer Problem Solver* ». DPS génère les mêmes points de position que GSP. Par contre, il peut placer des polygones. Les points de position générés ne correspondent pas à la projection des cotés des polygones mais du rectangle dans lequel chaque polygone est

31 La combinatoire apparente est définie comme le produit de la largeur des domaines de définition des variables entières.

inscrit. La Figure 4.2 illustre un exemple d'aménagement d'une salle de piano. *Pfefferkorn* a également proposé un test qui sert de *benchmark* et que nous reprendrons au *chapitre 6*. DPS tout comme GSP est limité par l'impossibilité de définir des domaines de valeurs aux variables, ainsi que par la non-complétude des solutions.

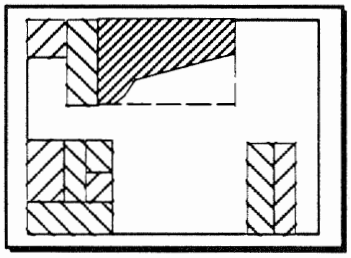


Figure 4.2 : Exemple d'aménagement d'une salle de piano [Pfefferkorn, 1975]

4.1.1.3. L'approche optimisation mathématique

Ligett [Ligett, 1981] a utilisé cette approche pour résoudre des problèmes de placement en architecture. La spécification du problème se fait à partir d'une matrice d'adjacence dans laquelle trois relations spatiales sont possibles : l'adjacence (représentée par la valeur *1*), la non-adjacence (représentée par la valeur *-1*) et « l'une ou l'autre » (représentée par le chiffre *0*). C'est à partir d'un algorithme d'optimisation que se fera la recherche de la solution numérique minimisant certains critères comme la surface totale de placement.

Dans l'approche proposée par Ligett, il n'est pas possible de spécifier une contrainte d'adjacence généralisée ou des contraintes disjonctives, telles qu'elles existent dans un cahier des charges d'un bâtiment. Il n'est pas effectué de réduction des domaines des variables géométriques lors de la spécification de la matrice d'adjacence ce qui rend impossible l'utilisation de cette approche pour la définition du *niveau topologique* d'ARCHiPLAN, lequel nécessite une réduction des domaines des valeurs des variables géométriques des différents espaces. Cette approche s'est heurtée également au problème de l'explosion combinatoire et aux contraintes peu extensibles. De plus, le minimum global n'est pas assuré.

4.1.1.4. L'approche système expert

André [André, 1986] a développé un système d'aide à l'aménagement spatial en architecture navale. Les contraintes développées sont déclenchées à partir de règles. *André* utilise des contraintes géométriques floues (*près-de, loin-de...*). L'algorithme d'énumération des solutions mis au point permet d'estimer l'importance relative des

contraintes floues au moyen de degrés de sévérité et ceci grâce à une expertise du domaine d'application, ce qui permet de hiérarchiser par ordre de priorité la résolution des contraintes floues.

Flemming [Flemming, 1988] a développé le système LOOS. LOOS se limite à quatre relations spatiales : *à l'est, à l'ouest, au nord* et *au sud*. Un problème de placement repose sur une représentation par graphe où les nœuds représentent les espaces et les arcs une des quatre relations spatiales déjà citées. La résolution du problème s'effectue en deux étapes. La première étape consiste à définir toutes les relations spatiales entre tout couple d'espaces ; aucune réduction des domaines des variables géométriques des espaces n'est effectuée au cours de cette étape. La deuxième étape consiste à tester si la configuration vérifie la cohérence des contraintes (relations spatiales) spécifiées dans le graphe ; si elle est cohérente alors l'instanciation des variables est effectuée. La représentation par graphe du problème ne correspond pas à celle d'un *schéma fonctionnel en architecture*, car il est impossible dans LOOS de représenter des contraintes d'adjacence généralisées ou des contraintes disjonctives. Par ailleurs, l'inexistence de réduction des domaines desvariables ne permet pas de considérer le *niveau topologique* tel qu'il est spécifié dans ARCHiPLAN.

Pour résoudre des problèmes de placement, *Kovacs* [Kovacs, 1991] a utilisé la programmation logique (Prolog) en utilisant le mécanisme d'unification. *Kovacs* n'a proposé que peu de contraintes fonctionnelles (cf. *chapitre 4*).

4.1.1.5. L'approche programmation par contraintes

Baykan et *Fox* [Baykan et Fox, 1991], *Maculet* et *Charman* [Charman, 1995 ; Maculet, 1991] ont utilisé des techniques de programmation par contraintes :

- *Baykan* et *Fox* ont développé le système WRIGHT. Ils effectuent une recherche dirigée par les contraintes « *Constraint Directed Search* » déjà utilisé par *Fox* dans son système d'ordonnancement *ISIS* (cf. *annexe A.3*). L'algorithme d'énumération des solutions mis au point est structuré en trois phases. Tout d'abord, il génère le graphe de contraintes, comme celui de la Figure 4.3. La complexité du problème est réduite en choisissant la décision la plus opportuniste. Cette décision se fait sur la base de critères comme le nombre de contraintes portant sur une variable, la dépendance des contraintes et la dureté des contraintes. Nous nous inspirons de cette approche pour proposer des heuristiques adaptées à notre problématique.

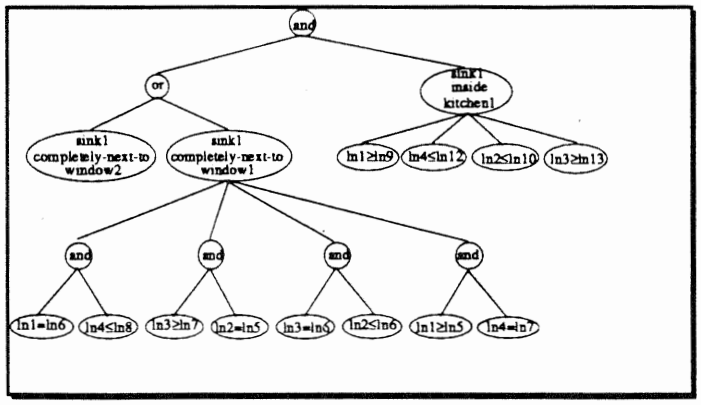


Figure 4.3 : Graphe partiel de contraintes généré par WRIGHT [Baykan et Fox, 1991]

- Dans ARCHIPEL, *Maculet* [Maculet, 1991] a utilisé des méthodes de propagation statique. Aucune propagation ni réduction des domaines des variables ne s'opère au cours de l'énumération. Un *backtrack* classique a également été utilisé.

- [Honda et al, 1994] ont utilisé les CLP (*Constraint Logic Programming)* pour résoudre des problèmes de placement en architecture.
  L'utilisation de la contrainte cumulative (cf. 3.4) développée par [Aggoun et Beldiceanu, 1992] au-dessus de CHIP (également un CLP) permet une réduction efficace des domaines des variables.

- Dans EAAS « Environnement d'Aide à l'Aménagement Spatial », *Charman* a proposé une nouvelle cohérence semi-géométrique plus adaptée à des problème spatiaux (cf. *chapitre 3*). Cette nouvelle cohérence procède au niveau des configurations des espaces plutôt qu'uniquement au niveau des paramètres, c'est-à-dire qu'elle raisonne en terme de réduction de domaine du rectangle ($x1$, $y1$, L, W) plutôt qu'en terme de réduction des domaines de $x1$, $y1$, L et W séparément. Il y a donc ici un couplage entre les réductions des domaines des paramètres des différentes pièces, un peu comme dans l'optique de la contrainte cumulative. Il s'agit du non-recouvrement des pièces que nous avons déjà illustré par la Figure 3.20 du *chapitre 3*.
  Plusieurs techniques de propagation ont été proposées, notamment celle du « *Check Forward* », celle du « *Real Full Look Ahead* » et celle du « *Check Backward* » [Haralick et al, 1980]. Le « *Check Forward* » consiste à avoir une propagation immédiate et partielle lorsqu'à chaque placement et dimensionnement d'une pièce, un filtrage sur les configurations des pièces non encore instanciées est effectué de façon à rendre, selon *Charman*, toutes les contraintes *semi-géométriquement cohérentes*. Le « *Real Full Look Ahead* » consiste à avoir une propagation globale. Dans le cas où aucun choix de propagation

n'est effectué, cela revient à faire du « *Check Backward* ».

Toutes les stratégies proposées aboutissent à l'énumération de solutions de placement que nous appellerons par la suite *solutions numériques*, ce qui, malgré la programmation par contraintes, génère toujours, en pratique, pour des problèmes de taille correcte une explosion combinatoire à cause du nombre même de solutions à énumérer (problème NP-complet). Nous résorberons ce problème avec l'énumération de solutions topologiques qui permettent de ne pas se focaliser immédiatement sur des détails d'ordre numérique, en privilégiant plutôt les principes de placement.

4.1.2. Les heuristiques

Les heuristiques de placement ont pour but d'obtenir un arbre de recherche (cf. Annexe 12) le plus élagué possible. Elle permettent de réduire considérablement les temps de réponse.

- Dans *GSP*, *Eastman* place d'abord les espaces les plus contraints. Dans *EPS*, *Pfefferkorn* place en premier les espaces de grande taille. Chez eux ces deux heuristiques sont d'ordre statique, c'est-à-dire que ces classements d'espace sont effectués une fois pour toutes au départ. Rien n'empêche d'utiliser ces heuristiques de manière dynamique.

- Dans les système plus évolués comme *CADOO*, *André* prend en compte des critères statiques comme la taille et le nombre de contraintes sur les éléments à placer, mais inclut des critères dynamiques comme le nombre de contraintes avec les espaces déjà placés.

- *Baykan* et *Fox* effectuent une recherche dirigée par les contraintes et non plus par les espaces comme précédemment. Les heuristiques introduites pour choisir la décision la plus opportuniste reposent sur des critères comme le nombre de contraintes portant sur une variable, la dépendance des contraintes et leur dureté.

- Dans EAAS, *Charman* a introduit l'heuristique pour "choisir le prochain coin". Le principe consiste à remplir les coins disponibles de l'espace de placement jusqu'à ce qu'ils soient tous remplis. A chaque placement d'un espace, de nouveaux coins apparaissent. Le choix du coin pour placer le prochain espace tient compte de critères tels que le nombre d'espaces pouvant y être placés, la distance du coin à l'origine ou encore du plus petit espace pouvant y être placé.

4.1.3. Notre réflexion sur ces travaux

Les stratégies générales de résolution s'orientent toutes vers des solutions numériques, elle permettent d'énumérer toutes les solutions ou les *n* premières. De plus, les exemples mis en œuvre ne concernent que des bâtiments à un seul étage (rez-de-chaussée). Sur le plan méthodologique, il n'est pas tenu compte de toutes les phases de conception préliminaire en architecture comme la phase d'esquisse. Sur le plan des stratégies de résolution, nous les trouvons insuffisantes pour résoudre des problèmes de grande taille (plus de 20 espaces, bâtiment à plusieurs étages...). Différentes techniques ont été observées, elles nous ont permis d'apprécier leurs performances et nous ont conforté dans notre choix des techniques de propagation de contraintes.

4.2. Algorithme d'énumération des solutions topologiques

La notion de solution topologique et l'algorithme d'énumération correspondant nous sont propres. Elles permettent, d'une part, de résorber le problème d'explosion combinatoire même pour des problèmes de grande taille (plus d'une vingtaine d'espaces) qui seraient sous-contraints. D'autre part, elle correspond à une étape réelle du processus de conception de l'architecte (cf. *chapitre 2*), ce qu'un bon outil d'aide à la conception se doit de faire. Dans notre esprit, une solution topologique correspond à une esquisse où les adjacences des espaces sont définies mais leurs dimensions restent assez imprécises, la précision géométrique n'étant abordée qu'à une étape ultérieure. La Figure 4.4 illustre un exemple d'esquisse montrant bien les différentes adjacences et laissant apparaître cette imprécision géométrique des différents espaces.

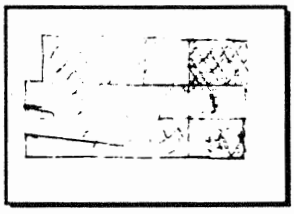


Figure 4.4 : Illustration d'une esquisse (dessin à main levée) de Le Corbusier (Villa de Carthage, 1928)

4.2.1. Définition d'une solution topologique

Lorsque l'instanciation des $\left[\frac{1}{2}n(n-1)\right]$ *variables d'orientation relative* émanant des *contraintes de non-recouvrement* s'avère a priori cohérente, l'état courant est potentiellement une *solution topologique*. A ce stade, les domaines de définition ont subi des réductions, on conçoit donc que peuvent exister plusieurs solutions

numériques relatives à cette solution topologique. Il est bon de rappeler que le modèle de contraintes (notamment les contraintes de symétrie, d'absorption de redondances, de non-recouvrement, d'adjacence généralisée et d'adjacence avec le contour) a été développé de manière à ce qu'une solution numérique ne puisse découler que d'une seule solution topologique. C'est ce principe qui a conduit aux partitionnements des orientations relatives.

Une solution topologique est définitivement confirmée après la validation d'au moins une solution numérique associée. Les *solutions topologiques* acquièrent ainsi la propriété de *cohérence*. Cette recherche d'une première solution numérique est souvent extrêmement rapide. Cette recherche utilise les mêmes heuristiques que l'algorithme d'optimisation des solutions numériques présenté à la section suivant. Une seule descente dans l'arbre de recherche (jusqu'à la première solution) est effectuée dans le premier cas.

Définition 4.1 : Une solution topologique est un ensemble contraint résultant de l'orientation relative de tout couple de pièces assortie d'au moins une solution numérique associée.

Remarque : Telle qu'on l'a défini, une solution topologique est cohérente au sens informatique du terme.

Définition 4.2 : Une solution numérique, dite *solution de placement*, est une instanciation de toutes les pièces satisfaisant les contraintes.

Propriété 4.1 : Une solution topologique est un ensemble de solution numériques (au moins une).

Propriété 4.2 : La notion d'orientation relative présente dans la définition d'une solution topologique est définie de telle manière que deux solutions topologiques soient disjointes. Une même solution numérique ne peut pas correspondre à deux solutions topologiques. On dit alors qu'une solution topologique constitue *une classe d'équivalence de solutions numériques*.

Il n'y a pas de nouvelle *variable d'orientation* relative entre *e1* et *e2* qui soit créée lorsque e1 et e2 sont contraints dans le schéma fonctionnel par une *variable d'adjacence* ou une *variable d'orientation relative*.

non-recouvrement (e1, e2) ≡ Adjacence (e1, e2, d1=[0,+∞], d2=[0, +∞]). (cf. §3.2.2.2.)

Très naturellement, nous avons désiré représenter graphiquement les *solutions*

topologiques pour voir si cela correspondait à notre idée d'une esquisse. Comme les variables, malgré une forte réduction de leur domaine, ne sont pas a priori instanciées, nous avons naturellement adopté pour la représentation graphique les valeurs moyennes des domaines des attributs (x1, y1, x2, y2) des espaces. Nous nous sommes alors aperçu de la très grande ressemblance avec les esquisses qu'établit un architecte en phase de préconception et qui correspondent à des dessins sans grande précision numérique, mais qui respectent globalement le cahier des charges. De la même façon qu'une esquisse, la représentation graphique d'une solution topologique présente de légers chevauchements de rectangles (cf. Figure 4.5).

Nous avons illustré dans la Figure 4.5 un exemple de solution topologique de deux espaces e1 et e2 adjacents et dont le cahier des charges est défini par la suite :

- domaine des variables des espaces e1 et e2 :
  - $e1.W \in [1,4]$, $e1.L \in [1,3]$, $e2.W \in [2,4]$ et $e2.L \in [2,4]$,
- E est une instance de la classe étage :
  - $E.W=5$ et $E.L=5$, $E.x1=0$ et $E.y1=0$,
- les contraintes :
  - e1 Dans E → $e1.W \in [2,4]$ et $e1.L \in [1,3]$,
  - e2 Dans E → $e2.W \in [2,4]$ et $e2.L \in [2,4]$,
  - e1 est Adjacent à e2.

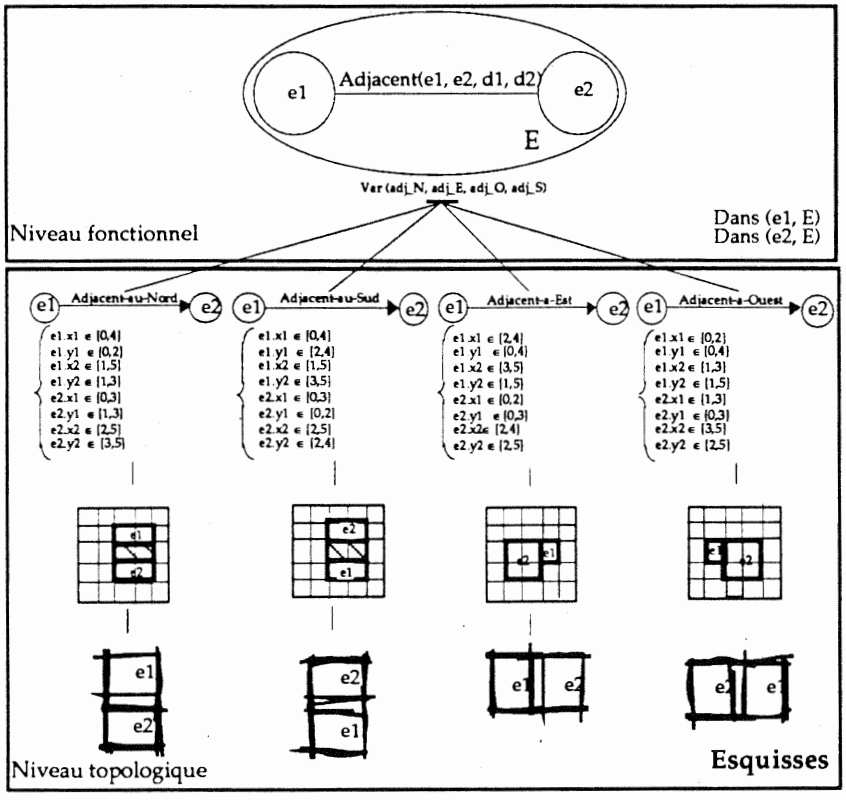


Figure 4.5 : Du schéma fonctionnel aux esquisses

Nous avons proposé un algorithme d'énumération topologique qui repose essentiellement sur une heuristique. Cette heuristique est originale puisque nous n'avons pas trouvé d'autres travaux en architecture sur les solutions topologiques telles que nous les définissons. L'heuristique introduite a été élaborée sur la base des travaux déjà cités mais surtout à partir d'une expertise des problèmes de placement que nous avons minutieusement abordés. Cette heuristique est présentée dans le chapitre suivant.

4.2.2. Heuristique d'énumération des topologies

Les approches de la programmation par contraintes qui ont vu le jour s'attelaient tout de suite à une énumération de solutions numériques, qui consistaient à choisir et à instancier successivement les espaces définis dans le cahier des charges. L'image qui était utilisée était de dimensionner puis de placer chaque espace dans l'espace de placement qui était initialement vide. Les heuristiques qu'ils ont proposées correspondaient essentiellement au choix du prochain espace à dimensionner et à placer [André, 1986 ; Eastman, 1973 ; Maculet, 1991 ; Pfefferkorn , 1975], et parfois au choix de l'endroit où placer en premier l'espace considéré [Charman, 1995].

Notre approche est totalement différente car il s'agit pour nous d'énumérer dans un premier temps des topologies. Cela ne correspond plus à l'instanciation d'un espace mais à celles des *variables topologiques*. Ce sont les variables d'*orientation relative* émanant des *contraintes de non-recouvrement* et des variables d'*adjacence relative* émanant des contraintes d'adjacence ainsi que des variables d'*adjacence relative avec le contour* émanant des contraintes d'adjacence avec l'espace de placement. Notre heuristique consiste donc à choisir l'ordre d'instanciation de ces variables. Néanmoins, notre algorithme consistera tout de même en une *détection* successive (et non plus un *choix*) d'espaces, pour lesquels nous instancierons les variables topologiques les référençant (celles qui n'ont pas déjà été instanciées).

Le principe de cette heuristique consiste, comme pour de nombreux problèmes utilisant la programmation par contraintes, à commencer par instancier les variables topologiques les plus contraintes de l'espace le plus contraint pour provoquer au plus tôt une propagation efficace et détecter dés que possible une incohérence, si elle existe. Cette heuristique est basée sur la détection de l'espace le plus contraint, qui correspond à l'espace possédant la plus grande valeur du *degré de contraintes*.

Initialement, l'attribut *deg-cont* de chaque espace est affecté d'une valeur au regard de ses contraintes d'adjacence avec l'espace de placement. Par exemple, un espace placé *Sur-la-façade-sud*, verra son degré de contrainte affecté de la valeur 4

(voir le tableau 4.1). Si deux espaces ont le même degré de contrainte, l'espace possédant la plus grande surface moyenne[32] sera *détecté* et si ça ne suffit pas pour les distinguer le premier de la liste sera *détecté*.

Contraintes	degré de contraintes
Adjacences de base avec l'espace de placement	4
Disjonction de deux adjacences de base	3
Disjonction de trois adjacences de base	2
Disjonction de quatre adjacences de base	1
conjonction de n adjacences de base	(4 x n)

Tableau 4.1 : Valeurs affectées au degré de contrainte au regard des contraintes d'adjacence avec l'espace de placement

Une fois *l'espace initial détecté* (l'espace le plus contraint à l'état initial avec l'espace de placement), on *place* en quelque sorte cet espace dans l'espace de placement en instanciant ses *variables topologiques*. A une étape donnée du processus *d'énumération topologique* pour laquelle un certain nombre d'espaces ont déjà été *détectés*, on *détecte* à nouveau un *espace courant* et on instancie ses *variables topologiques*. On procède jusqu'à ce que toutes les *variables topologiques* soient instanciées pour aboutir à une solution topologique potentielle, ou bien on effectue un *retour arrière* (*backtrack*) lorsqu'une incohérence est détectée pour rechercher d'autre solutions.

A l'issue de la *détection* du premier espace, il nous faut remettre à jour dynamiquement les degrés de contraintes des espaces restant à *détecter*. En considérant toujours les contraintes d'adjacence avec l'espace de placement, ces degrés de contrainte ne changeraient pas. Pour avoir une heuristique dynamique, nous avons utilisé les contraintes d'adjacence avec un *contour extérieur courant* qui est composé de tous les espaces déjà détectés. La mise à jour des degrés de contrainte des espaces restant à détecter se fait de manière incrémentale. Une seconde heuristique consistera à effectuer un classement des variables topologiques de l'espace courant détecté.

A l'état initial, l'espace de placement est vide, comme c'est indiqué dans le *cas (a)* de la Figure 4.6, le *contour extérieur courant* correspond à l'espace de placement. Une fois l'espace le plus contraint *détecté*, dans notre cas c'est l'espace *e1*. Le *contour extérieur courant* prendra la forme indiqué dans le *cas (b)*. A chaque état intermédiaire il y aura une modification du contour extérieur *et* une mise à jour des degrés de contraintes des espaces non détectés comme c'est indiqué dans le *cas (c)* et *(d)* de la même figure.

32. moyenne du domaine de valeur de la variable surface

Finalement, à chaque état intermédiaire, la *détection* d'un *espace courant* devra correspondre à celle de l'espace le plus contraint avec le *contour extérieur courant*. Par exemple seul l'espace e2 a une contrainte d'adjacence avec l'espace e1 donc avec le nouveau *contour extérieur courant*, à partir de là, le degré de contrainte de e2 sera incrémenté de la valeur 1 conformément aux valeurs associées à chaque type de contrainte indiquées dans le Tableau 4.2. Cette opération sera effectuée jusqu'à ce que tous les espaces soient détectés et leurs variables topologiques soient instanciées.

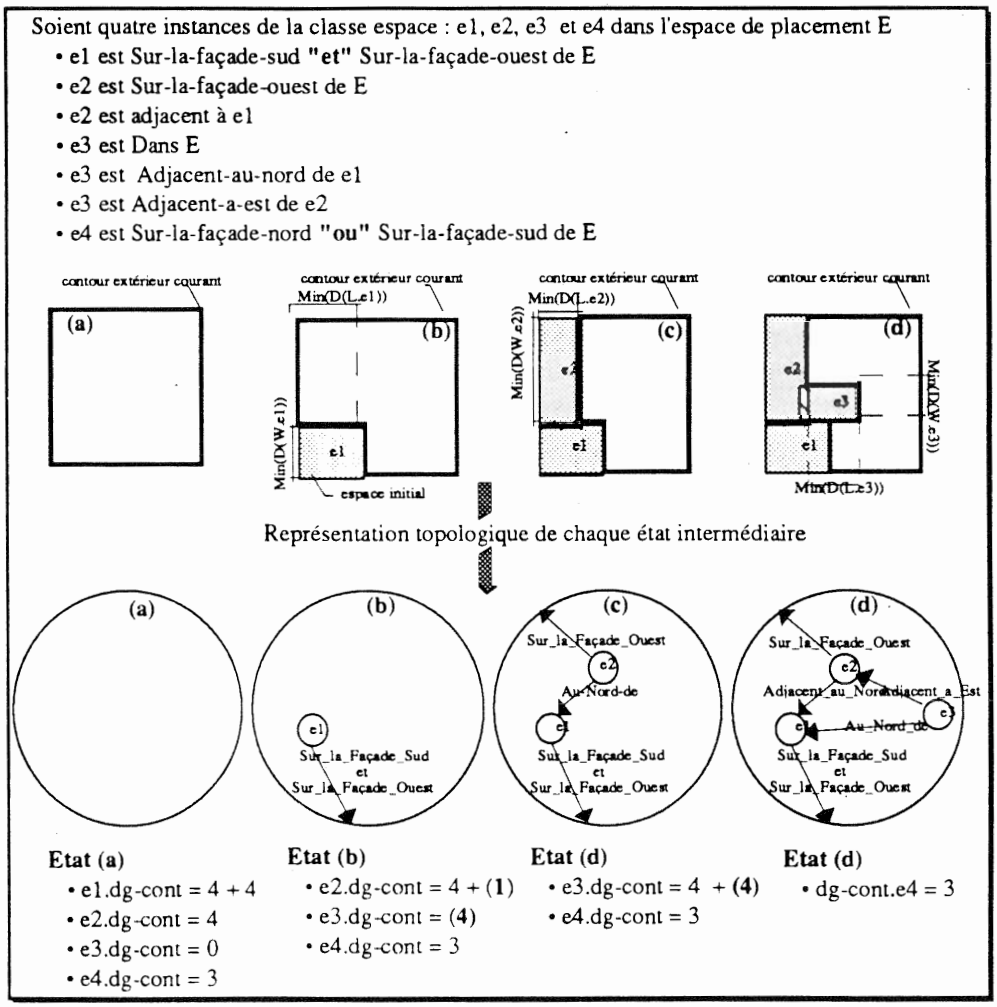


Figure 4.6 : Modifications dynamiques du *contour extérieur courant* lors de l'énumération topologique (les chiffres en gras représentent les valeurs ajoutées de manière incrémentale aux degrés de contraintes)

L'heuristique de classement des variables topologiques de *l'espace courant* restant à instancier, dépend par ordre préférentiel, des deux critères suivants :

- (1) le type de la variable : *variable d'orientation relative*, *variable d'adjacence relative* ou *variable d'adjacence relative au contour*. La priorité est donnée aux *variables d'adjacence*, car contrairement aux *variables d'orientation relative* elles contraignent beaucoup plus les espaces sur lesquels elles sont posées.

- (2) la largeur du domaine de chaque variable. Chaque variable est un point de choix, la priorité est donnée à celle qui présente le moins de choix.

Ces deux heuristiques correspondent bien aux heuristiques générales de la programmation par contraintes où le choix des premières variables est toujours porté sur les variables les plus contraintes.

4.2.3. Considération de temps de calcul

Le niveau topologique ainsi que l'heuristique dynamique du contour extérieur ont été très déterminants sur la réduction du temps de calcul. Pour valider cela, nous avons tout d'abord testé plusieurs problèmes proposés dans la littérature en comparant nos résultats avec ceux de Charman [Charman, 1995][33] (T-charman). Nous avons par ailleurs proposé de nouveaux tests beaucoup plus complexes. Les temps correspondants à la recherche de solutions sont exprimés en secondes CPU (sur IBM Risc 6000 320H). Nous avons illustré graphiquement les résultats de tous ces tests dans le *chapitre 6 (études de cas).*
Les exemples que nous avons testé sont les suivant :

- le problème du T2 : il s'agit de placer dans un rectangle de 9x4 deux rectangles de dimensions fixes 5x4 et 4x4.

- le problème du T4 : il s'agit de placer dans un rectangle de 9x9 quatre rectangles de dimensions fixes 5x4, 4x4, 5x5, 4x5.

- le problème de *Pfefferkorn* (Pfk) [Pfefferkorn,1975] :
  Il s'agit de placer dans un rectangle de 8x5 six rectangles de dimensions fixes 6x2, 4x2, 2x3, 2x3, 2x3 et 2x1. Les rectangles ont une seule orientation égale à 0°.

- le problème de *Laurière* (Lr) [Laurière, 1976] :
  Il s'agit d'une variante du problème de *Pfefferkorn*. Dans ce problème les rectangles peuvent tourner (deux orientations possible 0° et 90°), les autres données du problème demeurant inchangées.

- le problème de *Tong* (Tng) [Tong, 1987] :
  Il s'agit de placer dans un rectangle de dimension 9x9 quatre rectangles dont la longueur et la largeur peuvent varier entre 4 et 9.

---

33. Ayant reçu gracieusement de l'INRIA le prototype EAAS développé par [Charman, 1995], nous avons pu tester EAAS dans les mêmes conditions qu'ARCHiPLAN (même machine et en code Le-lisp interprété).

- le problème des 9 carrés parfaits (Col9) de *Charman* [Charman, 1995] inspiré de [Colmerauer, 1990] :
  Charman [Charman, 1995] a simplifié le problème de *Colmerauer* comme suit : il s'agit de placer, dans un rectangle de dimension 33x32, neuf carrés dont les côtés sont égaux respectivement à 1, 4, 7, 8, 9, 10, 14, 15 et 18.

- le problème de *Maculet* (Mac) [Maculet, 1991] :
  Il s'agit de concevoir le plan d'un logement F5. C'est le problème de placement de 10 espaces dans un espace de placement (cf. *chapitre 6*) pour plus de précisions sur les différentes contraintes du problème). Dans les travaux déjà effectués sur le placement, le problème de *Maculet* était le problème le plus complexe à traiter.

- le problème de l'ensemble de bureaux (Br) (nouveau "benchmark")
  Nous avons proposé ce test comme "benchmark" pour sa complexité (nombre d'espaces et de contraintes). Il s'agit de placer 16 espaces dans un espace de placement (cf. *chapitre 6.* pour plus de précisions sur les différentes contraintes du problème).

Le seul exemple représentatif de la conception en architecture et qui permet réellement de montrer la pertinence du niveau topologique d'ARCHiPLAN est le problème de *Maculet* car aucun attribut n'est instancié dés le départ et les contraintes sont de type fonctionnel.

EAAS de [Charman, 1995] énumère toutes les solutions numériques de chaque problème. En réalité, pour les problèmes où les espaces ont des dimensions fixes, nos solutions topologiques correspondent exactement aux solutions numériques. Par propagation de contraintes les domaines de valeur des points de référence *(x1,y1)* sont réduits à une valeur, grâce à la contrainte de *recouvrement total de l'espace de placement.* Pour les autres problèmes, nous avons énuméré toutes les solutions topologiques et ensuite toutes les solutions numériques pour chaque topologie, ce qui, répétons-le, ne correspond pas à la vocation d'ARCHiPLAN.

Dans un premier temps, nous avons comparé ARCHiPLAN et EAAS en désactivant l'absorption des solutions symétriques (cf. Tableau 4.2) puis en activant cette absorption (cf. Tableau 4.3).

Remarque : EAAS est en mode interprété. ARCHiPLAN est également en mode interprété mais utilise Pecos, qui lui est en mode compilé.

Problème	T2	T4	Tng	Pfk	Lr	Cl o9	Mac	Br
Nombre de pièces	2	4	4	6	6	9	10	16
Nbr Solutions	2	12	288	24	72	4	126	165
Temps (ARCHiPLAN)	0,093	1,53	11,89	3,46	78,5	4,92	3260	5840
Temps (EAAS)	xxx	xxx	236,4	23,76	xxx	12,45	5400	xxx

Tableau 4.2 : Influence de notre heuristique sur le temps de résolution (avec énumération des solutions symétriques)

Le principal constat est l'efficacité du *niveau topologique* et l'heuristique *h-dyn* aux tests effectués sur *EAAS* de [Charman, 1995]. Nous avons effectué les mêmes tests sans notre heuristique *h-dyn*, nous avons obtenu, pour chaque test, le même nombre de solutions indiqué dans le Tableau 4.2 avec des temps de réponse beaucoup plus importants.

Dans le Tableau 4.3 nous avons reporté le temps CPU des problèmes de Pfefferkorn, Laurrière et Tong sans les solutions symétriques. Là encore, nous constatons que nos temps sont plus courts que ceux obtenus dans EAAS [Charman, 1995].

Problème	Pfk	Lr	Tng
Nbr pièces	6	6	14
Nbr Solutions	4	12	12
Temps (ARCHiPLAN)	0,69	15,13	1,06
Temps (EAAS)	6,84	xxx	11,36

Tableau 4.3 : Influence de notre heuristique sur le temps de résolution (sans énumération des solutions symétriques)

Nous venons de montrer que sans avoir eu besoin de développer des *cohérences* hautement spécifiques comme la *cohérence semi-géométrique* de Charman, et sans utiliser les avantages, en termes de réduction de complexité, de notre niveau topologique, notre algorithme de placement est intrinsèquement meilleur que celui d'EAAS .

4.2.4. Vérification de cohérence d'une solution topologique

Une solution topologique n'est considérée comme telle que si elle est cohérente, c'est-à-dire s'il existe au moins une solution numérique (variables instanciées) qui en

découle. Ceci est fait en recherchant la première solution, en instanciant les variables géométriques des différents espaces sans procéder à une quelconque optimisation. Nous avons procédé à des tests pour évaluer la pertinence de la cohérence par arcs que nous évaluons en comparant le nombre de solutions topologiques potentielles (N1) et le nombre de solutions effectives (N2) (cf. tableau 4.4). Cette pertinence est évaluée par le pourcentage de solutions cohérentes parmi les solutions potentielles, soit $100 \times \frac{N2}{N1}$.

Problème	Tng	Pfk	Lr	Cl o9	Mac	Br
Nombre de pièces	4	6	6	9	10	16
N1 (nbr sol topo potentielles)	24	24	72	4	345	949
N2 (nbr sol topo cohérentes)	4	24	72	4	72	165
% de solutions cohérentes	17	100	100	100	21	18
T_sol_topo_potentielles (toutes)	0,65	11,1	72	4,56	3245	6225
Temps verif. cohérence (une sol)	0,085	0,09	0,09	0,090	0,200	0,15
T (recherche d'un sol optimisée)	0,103	0,111	0,120	0,110	0,205	0,18

Tableau 4.4 : Pertinence de la cohérence par arcs

Pour les exemples dont les longueurs et les largeurs sont instanciées, les solutions topologiques sont forcément cohérentes car ce sont aussi des solutions numériques. Dans le cas contraire, on constate que l'énumération topologique génère à peu près 5 solutions potentielles pour 1 solution cohérente. La cohérence par arcs nous semble donc une méthode suffisamment cohérente au sens informatique du terme. On remarquera que le temps de recherche de la solution numérique est sensiblement le même que le temps de recherche de la solution optimale. Ceci s'explique par le fait qu'une solution topologique correspond à un système fortement contraint. De ce fait, nous décidons de directement chercher la solution optimale d'une solution topologique plutôt que de rechercher une première solution.

4.3. Algorithmes d'optimisation des solutions numériques

Chaque solution topologique correspond pour l'architecte à un principe de placement. Énumérer toutes les solutions numériques de placement non seulement est illusoire, mais inutile pour l'architecte. Ce qui l'intéresse est de connaître la meilleure solution numérique correspondant aux principes de placement (solutions topologiques) qui ont retenu son attention. Cela correspond à l'étape de la coordination dimensionnelle du projet architectural (cf. *chapitre 2*). Pendant la

conception, l'architecte tient compte d'un certain nombre de critères (bonne orientation, coût, protection contre le vent, gain de surface dans les locaux habitables...) qui lui permettent d'obtenir de bonnes solutions ; il était donc pour nous nécessaire de clairement définir ces critères. Nous avons dressé une liste de critères qui nous semblent pertinents. Certes, ce ne sont pas les seuls, d'autres critères peuvent être proposés correspondant à un type d'application très particulier. Si un nouveau critère doit être introduit, ceci ne modifiera en aucun cas notre algorithme grâce aux techniques de programmation par contraintes qui découplent bien les contraintes et les algorithmes d'énumération et d'optimisation.

4.3.1. Critères d'optimisation

Lors de la conception d'un bâtiment, la question fondamentale que nous nous sommes posés était : quelles sont les fonctions désirées d'une conception architecturale, au sens de l'analyse fonctionnelle du besoin ? A partir de quelque travaux comme ceux de Goldschmidt [Goldschmidt, 1991] et Wiezel [Wiezel, 1991] et de notre propre analyse du processus de conception (cf. *chapitre 2*), nous avons dégagé la classification suivante des fonctions désirées :

- les fonctions artistiques (le parti pris architectural),
- les fonctions d'usage (ensoleillement, circulation, bruit...),
- les fonctions économiques (coût),
- les fonctions techniques (stabilité de la construction...).

Pour pouvoir tenir compte de ces critères, il est nécessaire de les formaliser. Le critère artistique ne sera pas abordé, parcequ'il dépend essentiellement du *libre-arbitre* de l'architecte, c'est un critère impossible à quantifier, il repose sur la culture, l'expérience et le vécu de l'architecte.

Peu de logiciels d'optimisation en architecture ont été développés jusqu'à présent, en phase préliminaire de conception. Nous citerons essentiellement le système développé par Ligett [Ligett, 1991].

L'optimisation d'une solution numérique correspondant à une solution topologique consiste à minimiser une fonction objectif ou « coût », notée C, qui est une combinaison pondérée des critères de fonctions élémentaires précédemment évoquées.

Le principe utilisé est celui, classique avec les environnements de programmation par contraintes, de « *Branch and Bound* », qui consiste à avoir une première solution de

coût C1, à poser immédiatement la contrainte statique (c'est-à-dire qui ne se défait pas par backtrack) : C < C1. La solution suivante sera donc forcément de coût moindre. Le fait de poser cette contrainte sur la fonction coût à chaque meilleure solution réduit les domaines ; une sorte de « réaction en chaîne » s'opère donc et la convergence vers le minimum s'accélère. Les heuristiques de choix d'une variable jouent sur la forme et la taille de l'arbre de recherche lors de l'énumération de toutes les solutions. Ce même arbre de recherche s'élague au fur et à mesure de son parcours lors d'une optimisation. A mêmes heuristiques de choix de variables, il est primordial, en optimisant uniquement, de trouver les meilleures heuristiques de choix de valeurs de manière à trouver immédiatement une première solution de coût C1 faible, pour avoir, par conséquent, une optimisation rapide.

4.3.1.1. La trame

Pour des raisons de coordination dimensionnelle en architecture (cf. *Chapitre 2*), il nous est nécessaire d'obtenir des solutions numériques s'inscrivant dans une trame (cf. Figure 4.7). Notre problème consiste donc à définir le critère (la variable contrainte) à optimiser afin d'obtenir des solutions dites « tramées ».

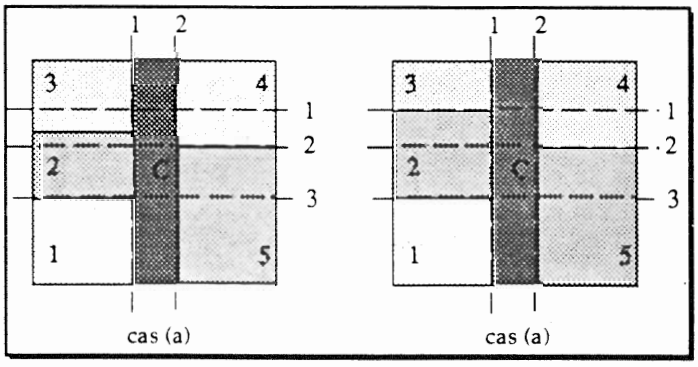


Figure 4.7 : Illustration pour une même topologie. Le cas (a) représente une solution numérique non tramée ; le cas (b) représente une solution tramée;

Le principe consiste à aligner le maximum de côtés des espaces sur les axes composant la trame. Cette optimisation revient à la minimisation du cardinal de l'ensemble des abscisses $\{x_i\}$ et de l'ensemble des ordonnées $\{y_i\}$ relatifs à la position des cloisons (les points de référence des espaces).

Ce critère est donc :

$$C\_trame = Card(\{x_i\}) + Card(\{y_i\})$$

Nous avons utilisé pour modéliser ce critère les contraintes ensemblistes disponibles dans la librairie PECOS.

4.3.1.2. Les surfaces de circulation

Habituellement un architecte cherche à réduire les surfaces de circulation au profit des locaux habitables. A cet effet nous avons développé le critère C_circulation défini par :

$$C\_circulation = \sum_{i=1}^{n} Circulation_i . S$$

Cette minimisation peut également être appliquée sur la surface d'un espace autre qu'une circulation.

4.3.1.3. La longueur des murs

Afin de minimiser le coût de la maçonnerie, nous nous devons de minimiser la longueur des murs et des cloisons. L'expression $\sum_{i=1}^{n} 2 \cdot (e_i \cdot L + e_i \cdot W) 2\ (e_i.L + e_i.W)$ représente la somme des périmètres de tous les espaces. Cette somme représente donc le périmètre de l'espace de placement plus deux fois la longueur des murs intérieurs car ils sont à chaque fois comptés deux fois, on adopte donc comme critère :

$$C\_Lmur = \sum_{i=1}^{n} (e_i.L + e_i.W) + \frac{E.L + E.W}{2}$$

Dans le cas où les dimensions de l'espace de placement sont instanciées dés le début, cela revient à minimiser la longueur des murs intérieurs. Lorsque les dimensions de l'espace de placement peuvent varier, le critère précédent tentera également de minimiser la longueur des murs externes. En fait la minimisation de longueur des murs influe sur le coût de construction. Il serait donc plus judicieux de considérer en relatif, le coût linéaire d'un mur externe par rapport au coût linéaire d'un mur interne. Ainsi, un critère plus pertinent serait :

$$C\_Lmur = \left[\sum_{i=1}^{n} (e_i.L + e_i.W) - \frac{E.L + E.W}{2} \right] \times Coût_{lin.interne} + (E.L + E.W) \times Coût_{lin.externe}$$

En réalité, ce critère est bon en première approximation, mais il faudrait tenir compte pour les murs internes de la part des murs porteurs. On voit donc que ce critère peut être complexifié à loisir.

Remarque : Il est démontrable que ce critère va privilégier les espaces d'aspect carré. Pour ce faire, prenons une pièce LxW de surface donnée unitaire LxW=1. L'évolution de L+W en fonction de L est régie par la fonction $y = L + \frac{1}{L}$ (cf. Figure 4.8) et montre que le minimum correspond au carré, pour lequel L = 1.

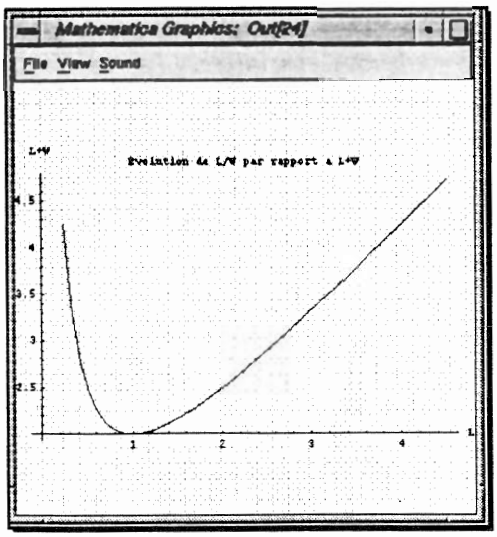


Figure 4.8 : A surface donnée, c'est le carré qui minimise la somme des côtés

4.3.2. Heuristique utilisée

Pour l'optimisation des solutions numériques à partir d'une solution topologique, l'heuristique introduite ne s'est pas avérée très importante car tous les espaces à instancier ont déjà vu le domaine de valeurs de leurs attributs géométriques fortement réduit. Une solution topologique est un système fortement contraint ne contenant plus de point de choix provenant des contraintes fonctionnelles. .

Après les différents tests (cf. Tableau 4.5) pour évaluer l'efficacité des différentes heuristiques connues dans la littérature, nous avons constaté que la meilleure heuristique consistait à instancier d'abord les espaces dont les points de référence sont les plus contraints (correspond à l'heuristique h_1 dans le Tableau 4.5). Ce sont ceux qui minimisent la surface maximale potentielle dans laquelle se situe le point (x1, x2). On privilégie donc les points qui minimisent :

$$[\text{Max}(D(x1))\text{-Min}(D(x1))] \times [\text{Max}(D(y1))\text{-Min}(D(y1))]$$

L'heuristique h_2 correspond au choix de l'espace dont le domaine de la surface est le plus grand. Cette heuristique est des plus classiques en placement. En effet, lorsqu'on part en vacances, il vaut mieux commencer à placer les bagages encombrants dans le coffre de sa voiture, puis boucher les recoins avec des petits paquets plutôt que le contraire... Comme nous le constatons dans le tableau 4.5, le gain de temps de h_1 par rapport à h_2 est très faible, de l'ordre du dixième de seconde (pour des problèmes de grande taille : plus de 20 locaux). Nous constatons également le temps très faible pour trouver une solution optimisée, ce temps est de l'ordre de la seconde pour le jardin

d'enfants.

Heuristique/Problème	Tng	Mac	R+1	JDenf
Temps pour h_1	0,103	0,576	0,678	0,756
Temps pour h_2	0,104	0,605	0,702	0,803

Tableau 4.5 : Le temps de recherche d'une solution optimisée pour une solution topologique avec l'utilisation de l'heuristique h_1 ou h_2 (les temps sont exprimés en secondes). R+1 correspond à l'exemple d'une maison individuelle et JDenf à celui d'un jardin d'enfant (cf. *chapitre 6*)

Afin d'obtenir les solutions optimales numériques de chaque solution topologique, il suffit d'appliquer l'algorithme classique du Branch and Bound avec cette heuristique pour chaque solution topologique.

4.4. Généralisation à plusieurs étages

Nous avons vu des algorithmes qui permettent d'énumérer des solutions topologiques et numériques optimales de bâtiments à un seul étage. Comme nous l'avons indiqué au *chapitre 2*, parmi nos principaux objectifs figurait la généralisation de notre application à des bâtiments à plusieurs étages.

4.4.1. Généralisation de l'algorithme d'énumération topologique

Généraliser l'algorithme d'énumération topologique à plusieurs étages revient tout simplement à *détecter* en premier les espaces de circulation verticale puis à appliquer en parallèle, l'algorithme d'énumération topologique. Ce parallélisme potentiel est actuellement traité de manière séquentielle un étage après l'autre. On peut envisager d'étendre la *détection* d'un espace courant à tous les espaces de tous les étages à la fois, c'est-à-dire de *paralléliser l'énumération topologique*. Il n'est pas certain que cette seconde solution soit la meilleure, car le backtrack risque de se déclencher seulement après la détection d'un certain nombre d'espaces sur un étage. Malgré tout, cette voie mérite d'être explorée.

4.4.2. Généralisation de l'algorithme d'optimisation numérique

Généraliser l'algorithme d'énumération d'optimisation numérique à plusieurs étages revient tout simplement à *instancier* en premier les espaces de circulation verticale puis à appliquer en parallèle, l'algorithme d'optimisation numérique. Comme l'énumération topologique, ce parallélisme potentiel est actuellement traité de

manière séquentielle un étage après l'autre. On peut, de même, envisager de *paralléliser l'instanciation* des espaces aux différents étages.

4.5. Conclusion

Nous avons, dans ce chapitre, présenté et expliqué notre propre démarche conduisant à l'énumération des solutions topologiques et à la recherche des meilleures solutions correspondantes. Nos algorithmes présentent plusieurs avantages :

- (1) **L'introduction de l'algorithme d'énumération topologique a permis d'obtenir des esquisses** qui, sur le plan méthodologique permettent à l'architecte d'apprécier les différentes solutions sans qu'il ait un nombre exorbitant de solutions numériques. Ce niveau de solutions topologiques est le cœur de notre apport puisqu'il a permis de découpler la problématique de calcul. En effet, on est passé d'une recherche d'une solution numérique optimale pour un problème avec des contraintes disjonctives à l'énumération d'un certain nombre de sous-problèmes fortement contraints avec des contraintes uniquement conjonctives, puis à la recherche des solutions numériques optimales pour chacun des ces sous-problèmes.

- (2) Nous avons sur le plan numérique introduit la **recherche des meilleures solutions en nous basant sur certains critères**. Actuellement, l'optimisation est faite sur un seul critère à la fois. Il faudrait à terme procéder à l'optimisation multicritères.

- (3) Le fait d'avoir la solution optimale (à partir de critères objectifs) pour chaque topologie permettra de connaître le coût minimal de chaque topologie. C'est là que les **fonctions artistiques**, difficilement codifiables, pourront être prises en compte lorsque l'architecte préférera tout de même privilégier une topologie moins optimale qu'une autre, pour des raisons de *parti pris*, mais en connaissance de causes.

- (3) Nous avons **généralisé ces algorithmes à des bâtiments à plusieurs étages**. Cette généralisation nous permet d'aborder un éventail très large de problèmes de conception en architecture. En procédant dans ces algorithmes à l'instanciation des circulations verticales d'abord, on pourra tester à l'avenir une parallélisation des algorithmes sur chaque étage.

Chapitre 5

Vers une CAO interactive en architecture

5.1. Introduction

Notre méthodologie de conception architecturale à trois niveaux : fonctionnel, topologique et numérique, nécessite de totalement repenser l'interface homme/machine de l'outil de CAO qui va la mettre en œuvre. Il faut notamment imaginer toutes les fonctionnalités d'un logiciel de CAO qui pourrait découler de cette approche. ARCHiPLAN est une première réponse à cette approche de conception architecturale. Nous proposons dans ARCHiPLAN (cf. Figure 5.1) :

- un éditeur de schémas fonctionnels,
- un gestionnaire de solutions topologiques,
- un gestionnaire de solutions numériques.

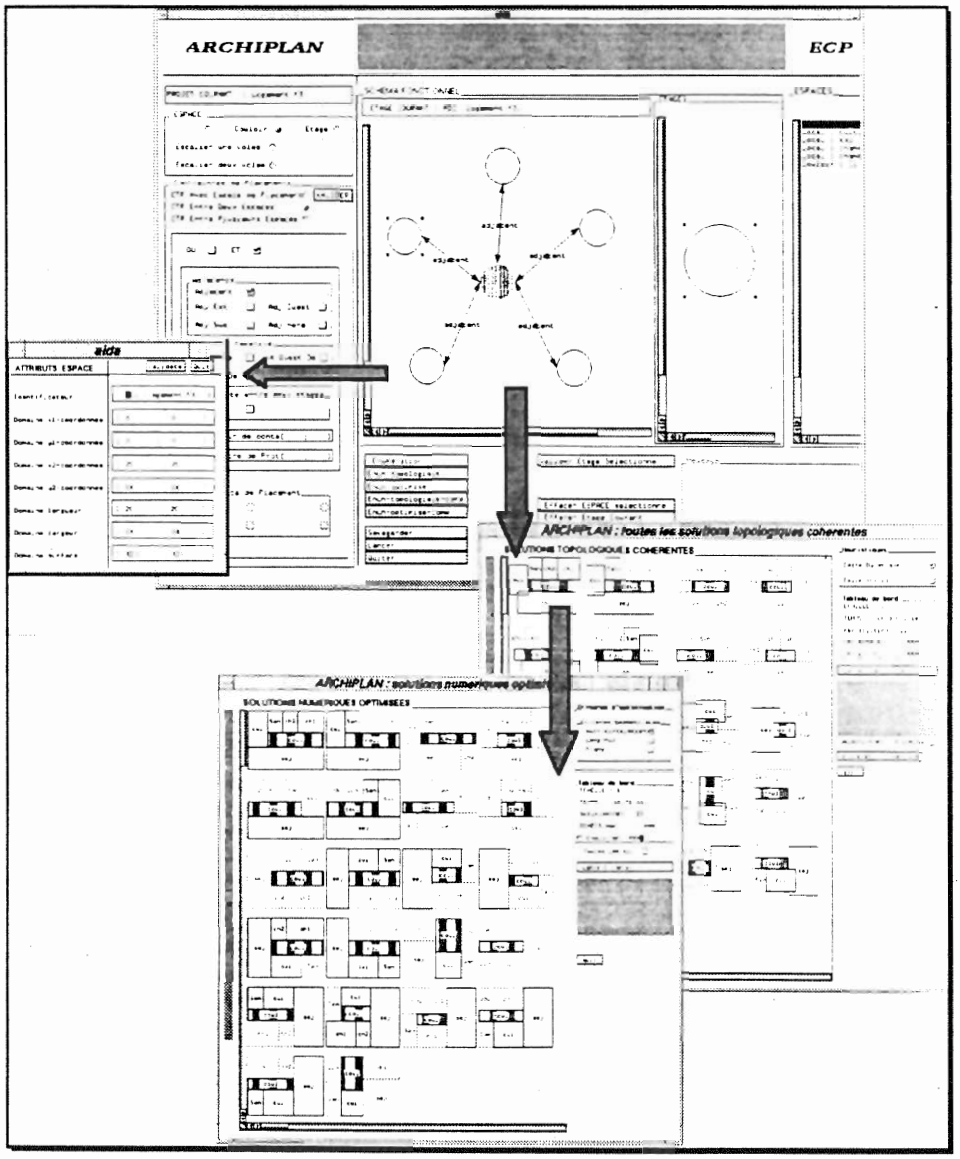


Figure 5.1 : Les trois principaux *panels* d'ARCHiPLAN correspondent aux trois niveaux : fonctionnel, topologique et numérique.

A travers l'exemple d'un logement F3, nous allons expliciter les différentes fonctionnalités d'ARCHiPLAN. Ce logement de 80 m^2 est décrit fonctionnellement par les contraintes dimensionnelles du Tableau 5.1 et les contraintes de placement relatif données par la suite. Le module correspond à 0,5 m, celle de surface à 0,25 m^2.

Locaux	Surface	longueur-Min	largeur-Min
•Séjour	72-128	6	6
•Cuisine	36-60	5	5
•SDB-wc	16-36	4	4
•Chambre1	36-60	6	6
•Chambre2	28-36	5	5
•Circulation	4-48	3	3

Tableau 5.1 : Contraintes dimensionnelles des espaces du logement F3

Les contraintes de placement prises en compte sont :

- les contraintes d'adjacence de tous les locaux avec le couloir pour un minimum d'un mètre de longueur de contact,
- l'adjacence entre la cuisine et la SDB (pour minimiser les longueurs de canalisations),
- l'adjacence entre le séjour et la cuisine (pour une meilleure desserte du séjour),
- l'orientation du séjour sur la façade sud de l'espace de placement (pour un meilleur ensoleillement).

5.2. L'éditeur de schémas fonctionnels

A partir du cahier des charges (programme du logement), l'utilisateur d'ARCHiPLAN pourra définir, à l'aide d'un éditeur, le schéma fonctionnel (organigramme) de son logement. Ce schéma est un graphe où les noeuds représentent les espaces du cahier des charges et les arcs les contraintes fonctionnelles entre ces espaces. Comme nous l'avons illustré dans la Figure 5.2, notre éditeur se compose de :

- un éditeur d'espaces,
- un éditeur de contraintes,
- deux fenêtres de représentation : l'une pour les étages, l'autre pour le contenu de l'étage courant.

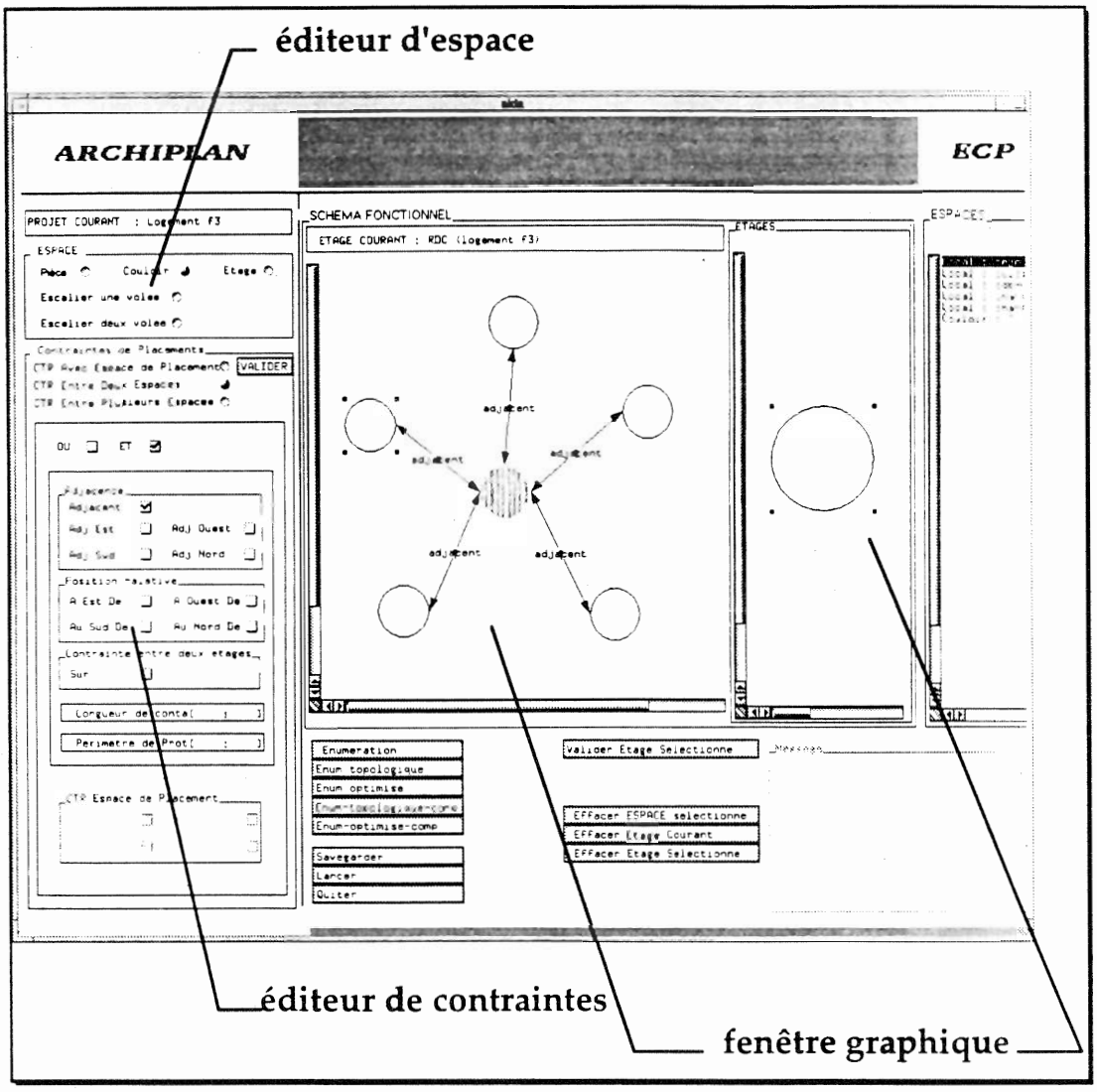


Figure 5.2 : L'éditeur de schémas fonctionnels. Il existe deux fenêtres graphiques correspondant respectivement aux contraintes entre étages et au contenu de l'étage courant.

5.2.1. Le graphe fonctionnel

La représentation du schéma fonctionnel dans ARCHiPLAN (cf. Figure 5.8) se fait sur deux niveaux. D'abord on représente les étages et leurs contraintes relatives. Dans notre cas, nous avons un seul nœud représentant l'espace de placement. Quant aux différents espaces, ils seront représentés dans la fenêtre graphique de gauche et seront tous considérés par défaut « dans » l'espace de placement. La conception en deux niveaux nous permet d'étendre notre méthode à plusieurs étages ou de travailler juste au niveau supérieur correspondant à celui du « Plan de Masse ».

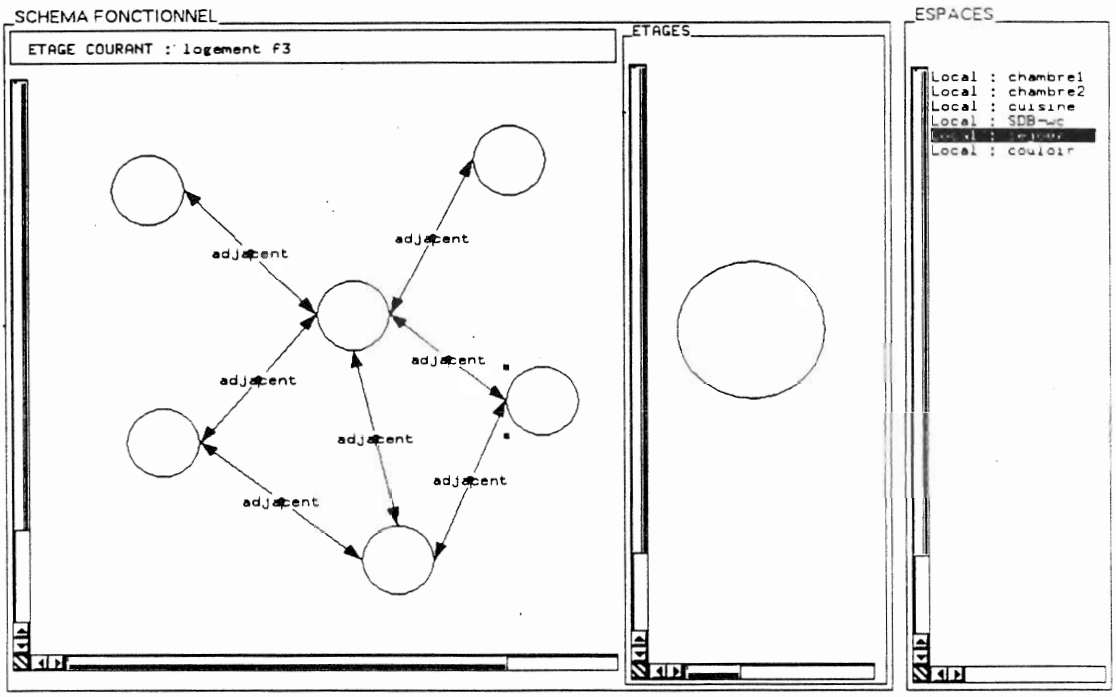


Figure 5.3 : Schéma fonctionnel du logement F3 dans ARCHiPLAN

5.2.2. L'Éditeur d'espaces

C'est à partir de cet éditeur (cf. Figure 5.4) que le concepteur pourra sélectionner le type d'espace à générer. Les différents types d'espace correspondent à ceux définis dans le modèle de connaissances architecturales (cf. *chapitre 3*).

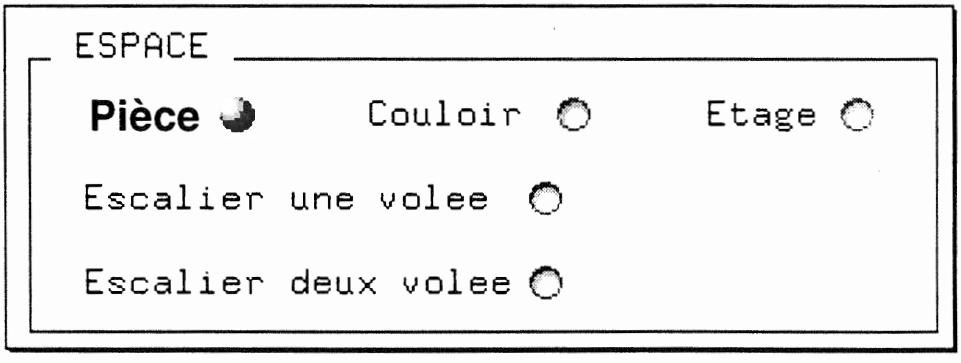


Figure 5.4 : L'éditeur d'espace

La sélection et la validation d'un espace permettent de générer parallèlement une instance de cet espace et un nœud représentant cette instance au niveau du graphe. Comme indiqué dans la Figure 5.4, seuls les étages (espaces de placement) peuvent être générés dans la fenêtre graphique de droite, la fenêtre de gauche étant réservée à la représentation des espaces contenus dans l'espace de placement.

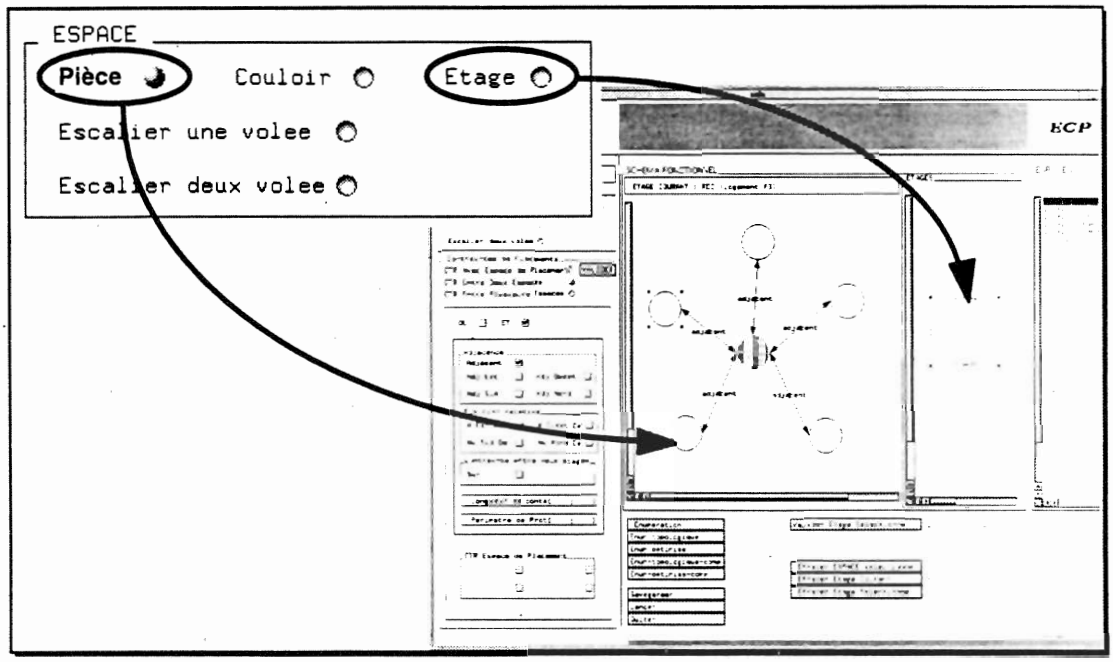


Figure 5.4 : Génération d'espace à partir de l'éditeur d'espaces

5.2.3. Les éditeurs de contraintes

Il existe deux éditeurs de contraintes : un éditeur pour les contraintes unaires et un autre pour les contraintes n-aires.

5.2.3.1. L'éditeur de contraintes unaires

L'éditeur de contraintes unaires (cf. Figure 5.6) permet d'entrer les domaines de valeur des différents attributs d'un espace (surface, longueur, largeur, point de référence, etc), sous la forme d'un intervalle.

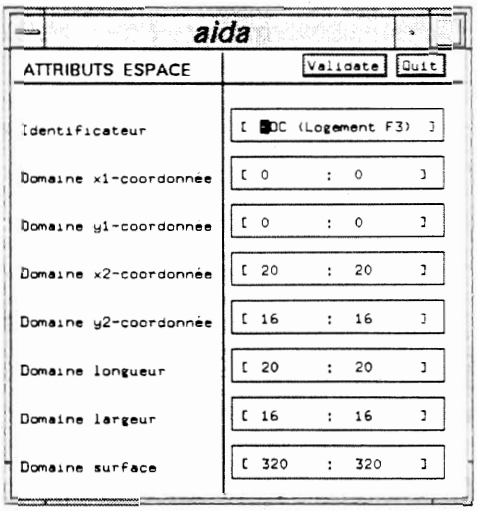


Figure 5.5 : Éditeur de contraintes unaires. Cet exemple concerne les domaines de valeur de l'espace de placement du logement F3

Comme nous l'avons indiqué au *chapitre 3*, il n'est pas nécessaire d'introduire les valeurs de tous les attributs d'un espace ; *L* et *W*, par exemple, peuvent suffire . Nous obtenons automatiquement et par propagation de contraintes, grâce aux contraintes de classe (cf. *chapitre 3*), les domaines de valeur de la surface et des coordonnées des points $(x1,y1)$ et $(x2,y2)$. S'il nous arrive de modifier une de ces valeurs, cette modification sera propagée. Chaque espace généré verra également le domaine de ses variables géométriques réduit automatiquement grâce à la contrainte d'inclusion dans l'espace de placement.

5.2.3.2. L'éditeur de contraintes n-aires

Les contraintes n-aires sont éditées à partir de la *zone de sélection* indiquée dans la Figure 5.7. Cette zone regroupe les contraintes avec l'espace de placement (*Sur-la-façade-Sud, Sur-la-façade-nord*, etc), les contraintes d'adjacence généralisée entre deux espaces (cuisine *adjacente* à la salle de bain) qui prend en compte le périmètre de protection et la longueur de contact, les contraintes de position relative et enfin les opérateurs de composition ET et OU (wc adjacent à la cuisine « OU » à la salle de bain, etc). La composition de contrainte est de la forme ((*e1* cont1 *e2*) OU (*e1* cont2 *e3*)) (cf. *chapitre 3*) ; il est possible de définir par exemple qu'*un espace e1 est adjacent à e2 avec une longueur de contact l1 OU e1 est adjacent à e3 avec une longueur de contact l2*. Les notions de *longueur de contact* et de *périmètre de contour* s'avèrent très puissantes à l'usage. Par exemple, la contrainte de non-adjacence est définie comme une contrainte d'adjacence avec une valeur 1 comme valeur minimale du périmètre de protection. Par défaut, nous avons pour la longueur de contact : $\text{Min}(D(d1))=0$ et $\text{Max}(D(d1))=+\infty$, et pour le périmètre de protection $\text{Min}(D(d2))=0$ et $\text{Max}(D(d2))=0$. Ceci correspond à une adjacence classique sans communication particulière.

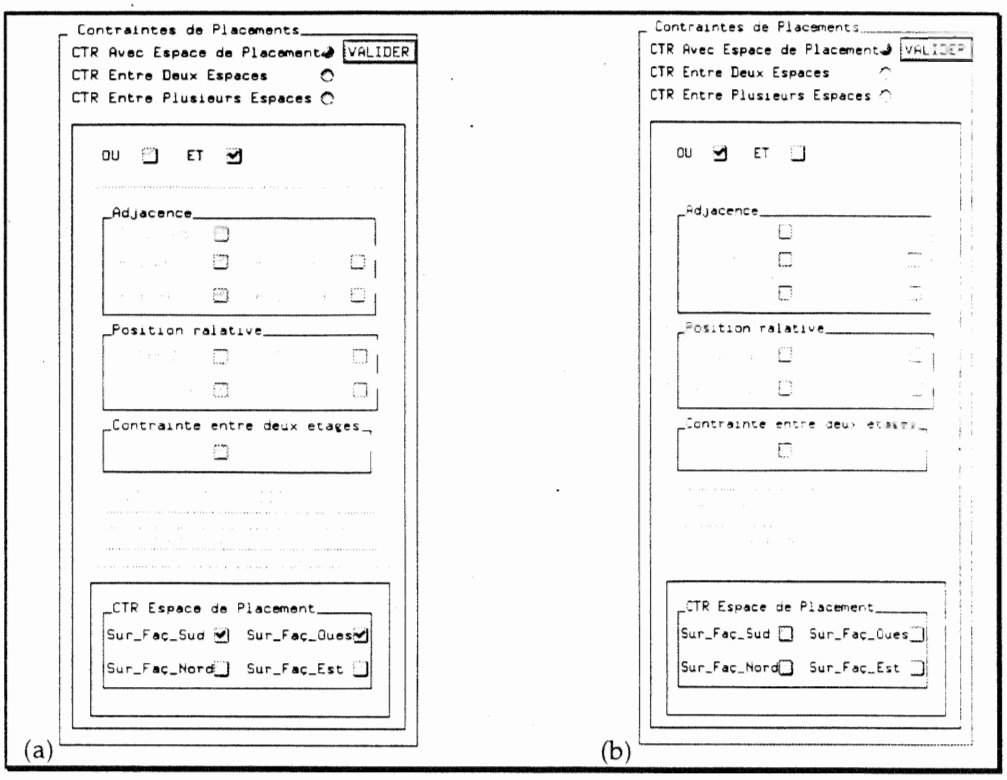


Figure 5.7 : Zone de sélection des contraintes n-aires dans ARCHiPLAN. Le cas (a) représente la contrainte *Sur-la-façade-sud* « ET » *Sur-la-façade-ouest* ; le cas (b) représente la contrainte *Sur-la-façade-sud* « OU » *Sur-la-façade-ouest*

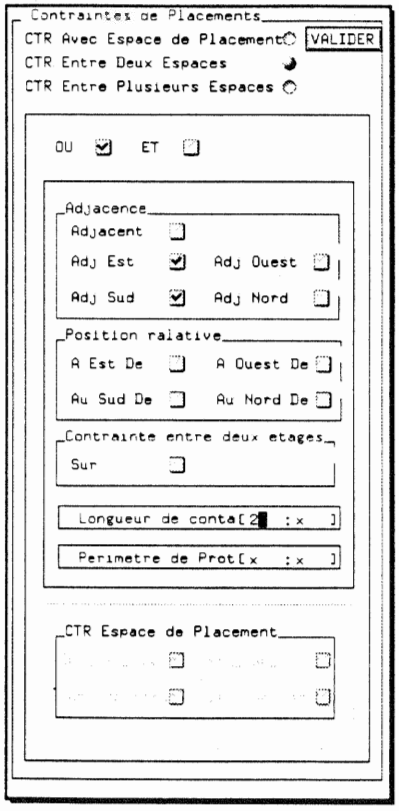


Figure 5.8 : Sélection de la contrainte » *Adjacent-a-est* « OU » *Adjacent-a-ouest* avec un minimum d'un mètre de longueur de contact.

A partir du panel illustré dans la Figure 5.7 on peut déclarer des contraintes élémentaires, on peut aussi entrer les compositions très répandues (utilisées surtout pour les circulations) de la forme ((*e1* cont1 *e2*) OU (*e1* cont2 *e3*). Selon que e2 et e3 représentent le même espace, deux ou trois "clics" sur les nœuds du graphe seront nécessaires. Lorsqu'on a affaire à trois espaces différents, il faut, entre le deuxième et le troisième "clic", entrer les domaines de d1 et d2 si la contrainte relative à e3 est une contrainte d'adjacence.

Dès qu'une contrainte est définie, elle apparaît graphiquement comme un arc du graphe (ou plusieurs arcs d'une composition à trois espaces). Toutes ces contraintes sont modifiables (c'est-à-dire rééditables) après une désignation graphique avec le bouton milieu de la souris (le bouton gauche sert au déplacement des entités du graphe).

Les contraintes implicites sont automatiquement prises en compte, il s'agit de contraintes de non-recouvrement, d'inclusion, de symétrie (elles sont détectées avant une énumération topologique) et de contraintes plus spécifiques (cf. *chapitre 4*). Ces contraintes peuvent être éventuellement désactivées, notamment la contrainte de recouvrement total de l'espace de placement ce qui permettrait d'avoir des solutions avec des « *espaces perdus* ». Dans ce cas, le critère de minimisation des surfaces de circulation s'étend à la prise en compte de la surface des espaces perdus : $\text{Critère} = E.S - \sum_{i=1}^{n} e_i.S$, les e_i étant les pièces habitables (sans les circulations).

5.3. Le gestionnaire de solutions topologiques

Tout comme un architecte dessine des esquisses à partir d'un cahier des charges, ARCHiPLAN génère toutes les solutions topologiques découlant du schéma fonctionnel et les représente géométriquement sous forme d'esquisses (cf. Figure 5.9). Le logiciel de CAO doit permettre d'appréhender le plus rapidement « *le champ du possible* » c'est-à-dire toutes les solutions topologiques et leurs spécificités propres. Ceci est absolument nécessaire au concepteur pour prendre la décision d'étudier plus en détail certaines topologies. Ainsi, il a été développé des fonctionnalités pour :

- **Comparer des solutions topologiques** deux à deux (entre une solution et la précédente) et mettre en évidence des différences topologiques entre les solutions successives grâce à des codes de couleurs : dans la Figure 5.10, nous distinguons une différence de couleur entres les deux solutions (*ch1* et *ch2*). Cette distinction met en valeur les différences topologiques entre la solution courante et la solution précédente.

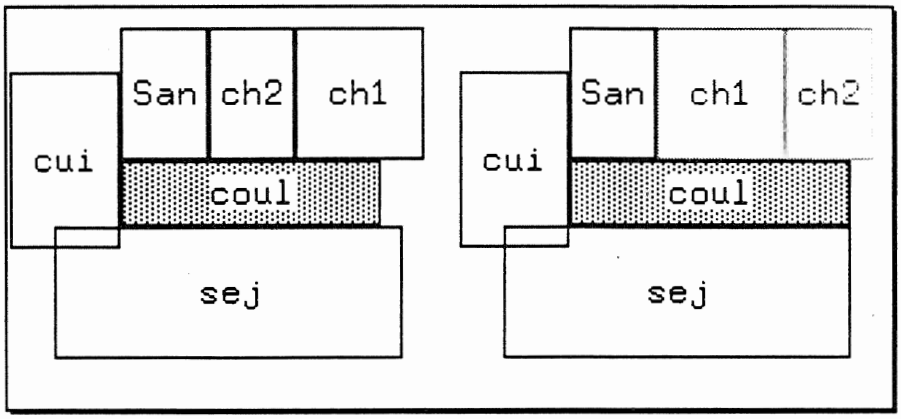


Figure 5.9 : Mise en évidence des différences topologiques grâce à des codes de couleurs

- **Trier des solutions topologiques** selon divers critères. Par exemple, on désire ne visualiser que les solutions topologiques qui laissent espérer la possibilité d'avoir une surface inférieure à 20 m^2. On pose alors la contrainte S<20 pour toutes les solutions topologiques, la propagation s'effectue, certaines solutions topologiques deviennent déjà incohérentes. On effectue enfin la vérification de cohérence en recherchant une première solution numérique. Finalement, le tri va consister à ne garder que les solutions topologiques initiales (avant la contrainte S<20) qui se sont avérées cohérentes après la pose de la contrainte S<20.

Le but de ces deux premières fonctionnalités est de permettre au concepteur de filtrer les topologies qui l'intéressent de manière à poursuivre l'étude (l'optimisation) pour ces seules solutions.

- **Appréhender une solution topologique**

Le concepteur peut bénéficier d'informations très importantes au niveau de chaque solution topologique en éditant les espaces d'une solution topologique pour avoir une idée de la réduction des domaines des variables géométriques.

- **Explorer numériquement une solution topologique**

ARCHiPLAN laisse la possibilité au concepteur d'aller « *manuellement* » explorer les différentes solutions numériques qui peuvent en découler, et ceci de manière séquentielle par réduction d'intervalles ou instanciations successives. A l'issue de chaque choix du concepteur, le logiciel opère une vérification de cohérence en cherchant une première solution numérique. Cette fonctionnalité de conception incrémentale par raffinements raisonnés est permise par la méthode même de programmation par contraintes. Elle n'existe pas en CAO variationnelle où on sait simplement que la géométrie est, à un moment donné, sous-contrainte, sans connaître les degrés de liberté restants.

De plus, on peut à un moment donné décider d'énumérer toutes les solutions numériques (avec le risque de l'explosion combinatoire). Les solutions numériques sont alors collectées et visualisables dans un *collecteur de solutions numériques* que nous présentons par la suite.

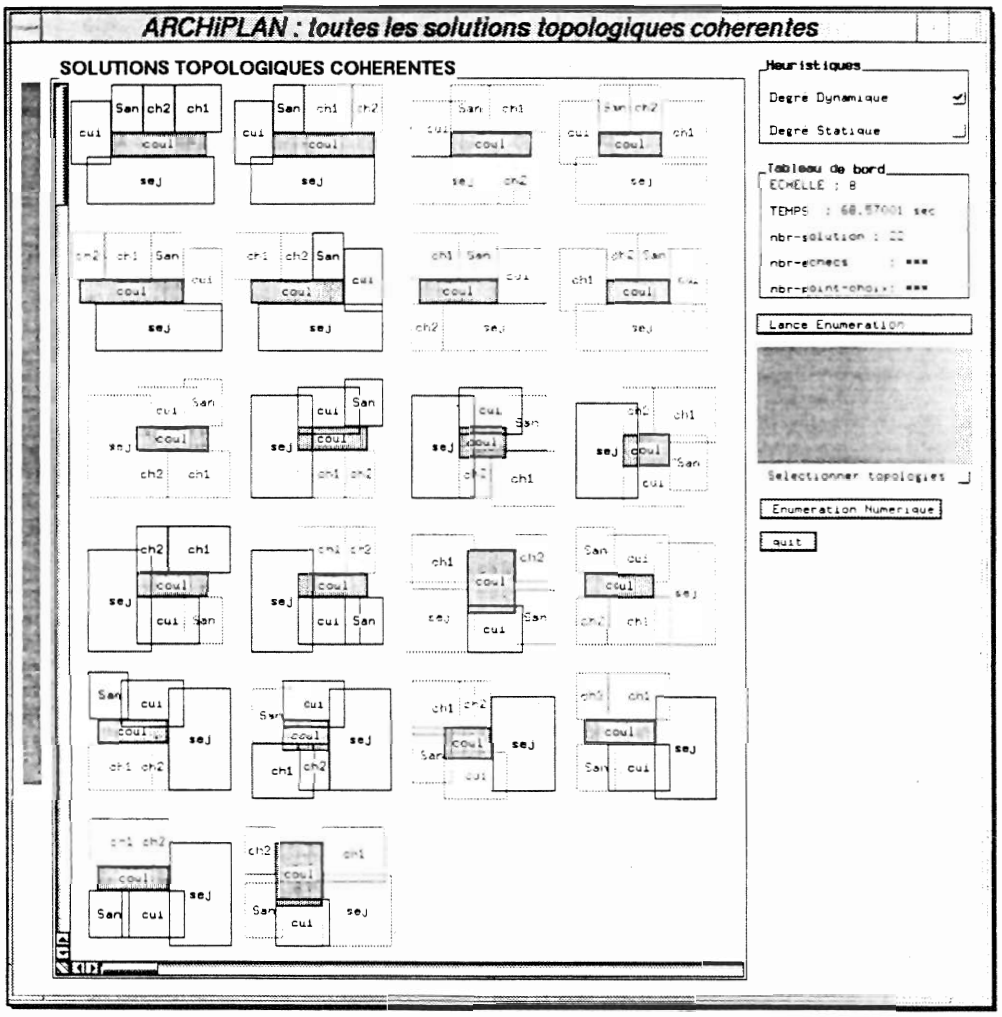


Figure 5.10 : Représentation des 22 solutions topologiques cohérentes du (F3)

5.4. L'éditeur de fonctions d'optimisation

Une fois les solutions topologiques obtenues et un certain nombre d'entre elles choisies pour une étude plus poussée (par défaut, toutes), le concepteur pourra sélectionner le critère de son choix afin d'obtenir les solutions numériques optimisées. A ce jour l'optimisation est mono-critère et seuls trois critères sont proposés (cf. Figure 5.11).

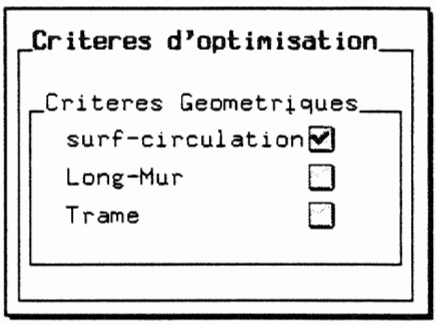


Figure 5.11 : Choix du critère minimisant la surface totale de circulation

5.5. Le collecteur de solutions numériques

Pour chacune des solutions topologiques auxquelles le concepteur s'intéresse, on recherche la *meilleure* solution numérique pour satisfaire au mieux le critère choisi tel que : minimiser la surface de circulation, solution tramée, minimiser la longueur des murs... La Figure 5.12 représente les solutions optimales correspondant aux solutions topologiques à partir du critère de minimisation de la surface de circulation sélectionnée.

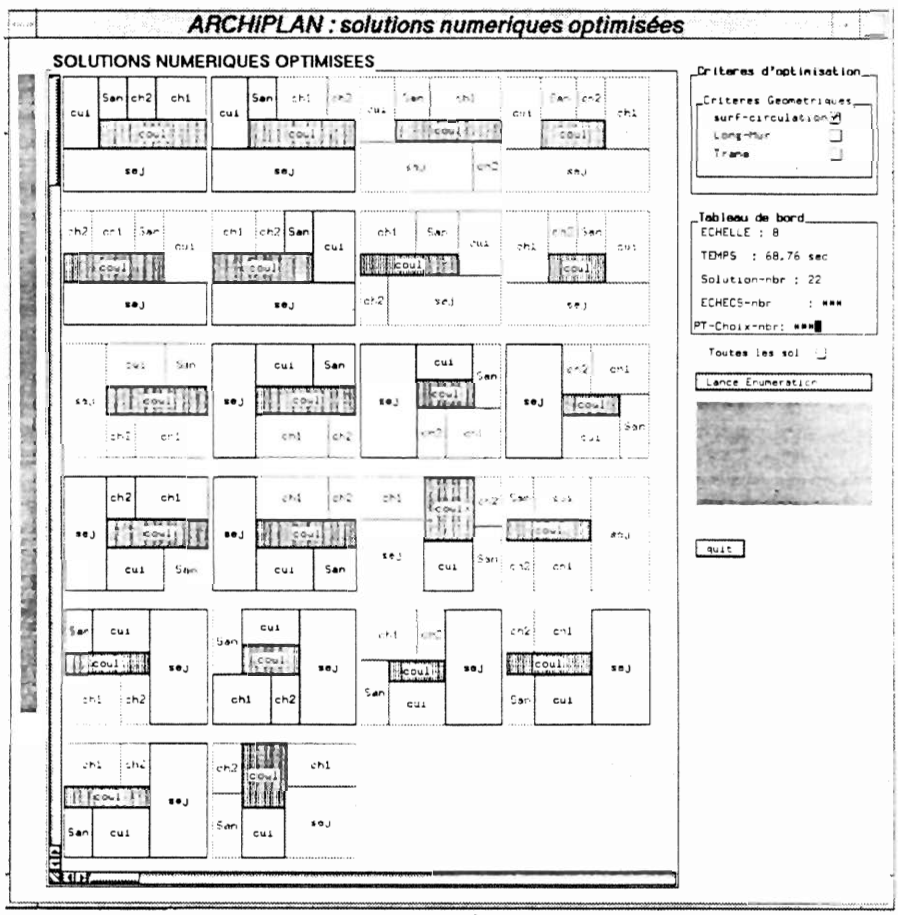


Figure 5.12 : Solutions numériques optimisant le critère de surface de circulation et relatives aux solutions topologiques de la Figure 5.9

Un ensemble de fonctions de gestion des solutions numériques est proposé :

- fonction d'ordonnancement des solutions numériques par ordre croissant de coût de la fonction d'optimisation,
- visualisation des différences topologiques entre solutions successives en utilisant des codes de couleurs,
- tris par rapport à une combinaison de conditions (requêtes de type base de données relationnelle). Le tri est plus simple sur des valeurs instanciées que sur les attributs des solutions topologiques.
- à ce stade, une seule solution numérique de coût minimal C*i* a été déterminée pour chaque solution topologique *i*. Il est donc laissé la possibilité pour une solution topologique de coût *Ci* d'énumérer toutes les solutions de coût minimal *Ci*. Là encore, la programmation par contrainte permet d'effectuer cette recherche très simplement. Il suffit de repartir de la solution topologique et d'ajouter *C=Ci* puis, d'énumérer toutes les solutions numériques. Dans le cas où il y a plus d'une solution, les solutions sont accessibles et se succèdent par une barre de défilement qui apparaît sous la solution numérique correspondante. Par « double-clic » sur cette barre de défilement, on accède à un nouveau collecteur de solutions numériques qui présente toutes les solutions numériques de coût *Ci* minimal, à la fois. On peut également utiliser les fonctions de tri sur ces solutions de coût Ci.

5.6. Conclusion

Dans ce chapitre nous avons présenté l'interface utilisateur d'ARCHiPLAN. Cette interface a été élaborée avec le souci d'offrir les fonctionnalités les plus pertinentes en conception architecturale. Cette interface s'est avérée extrêmement souple et efficace. Il est certain que nous n'avons pas encore fait le tour de toutes les fonctionnalités qui peuvent découler de la programmation par contraintes.

Chapitre 6

Études de cas

Ce chapitre est consacré à la validation d'ARCHiPLAN par des études de cas. Ce ne sont pas les exemples qui manquent pour valider ARCHiPLAN car ARCHiPLAN s'avère, de par le niveau topologique, très robuste vis-à-vis de problèmes de taille moyenne et de problèmes sous-contraints.

Par contre, nous pouvons constater que la plupart des "benchmarks" proposés dans la littérature (sauf celui de *Maculet* [Maculet, 1991]) ne sont pas représentatifs de problèmes réels de conception en architecture, car la plupart du temps les dimensions (L et W) des espaces sont instanciées dés le départ. Ceci est dû au fait que peu de systèmes peuvent résoudre de manière pertinente (temps, explosion de solutions) le problème dans toute sa généralité (L, W et S définis sur des domaines). Les "benckmarks" de la littérature ne permettent pas montrer toute la pertinence d'ARCHiPLAN car nous avons vu que pour les problèmes où L et W sont instanciés dès le départ, une solution topologique est directement instanciée en une solution numérique. Néanmoins, nous montrons que l'approche d'ARCHiPLAN est déjà la meilleure. Nous proposons par la suite quatre nouveaux "benchmarks".

6.1. Problèmes déjà proposés comme "benchmarks"

6.1.1. Le problème de Pfefferkorn

Il s'agit de placer dans un rectangle de dimension 8x5 six rectangles de dimensions fixes 6x2 (espace 1), 4x2 (espace 2), 2x3 (espace 3), 2x3 (espace 4), 2x3 (espace 5) et 2x1 (espace 6). Les rectangles ont une seule orientation égale à 0°.

- **Solutions numériques**

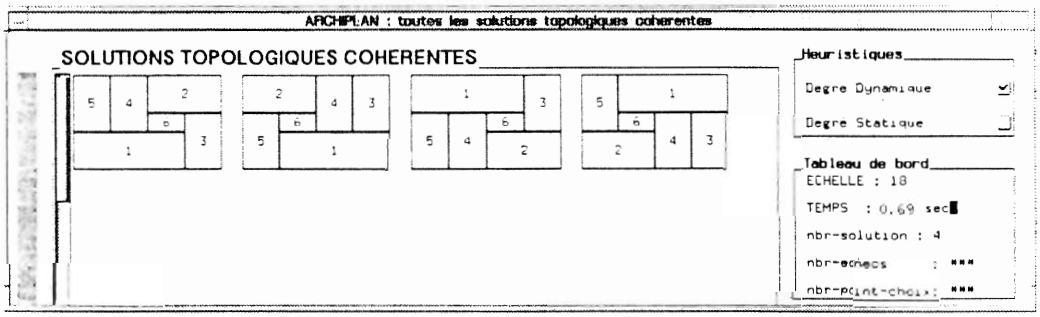


Figure 6.1 : Les 4 solutions du problème de *Pfefferkorn* (sans les solutions symétriques)

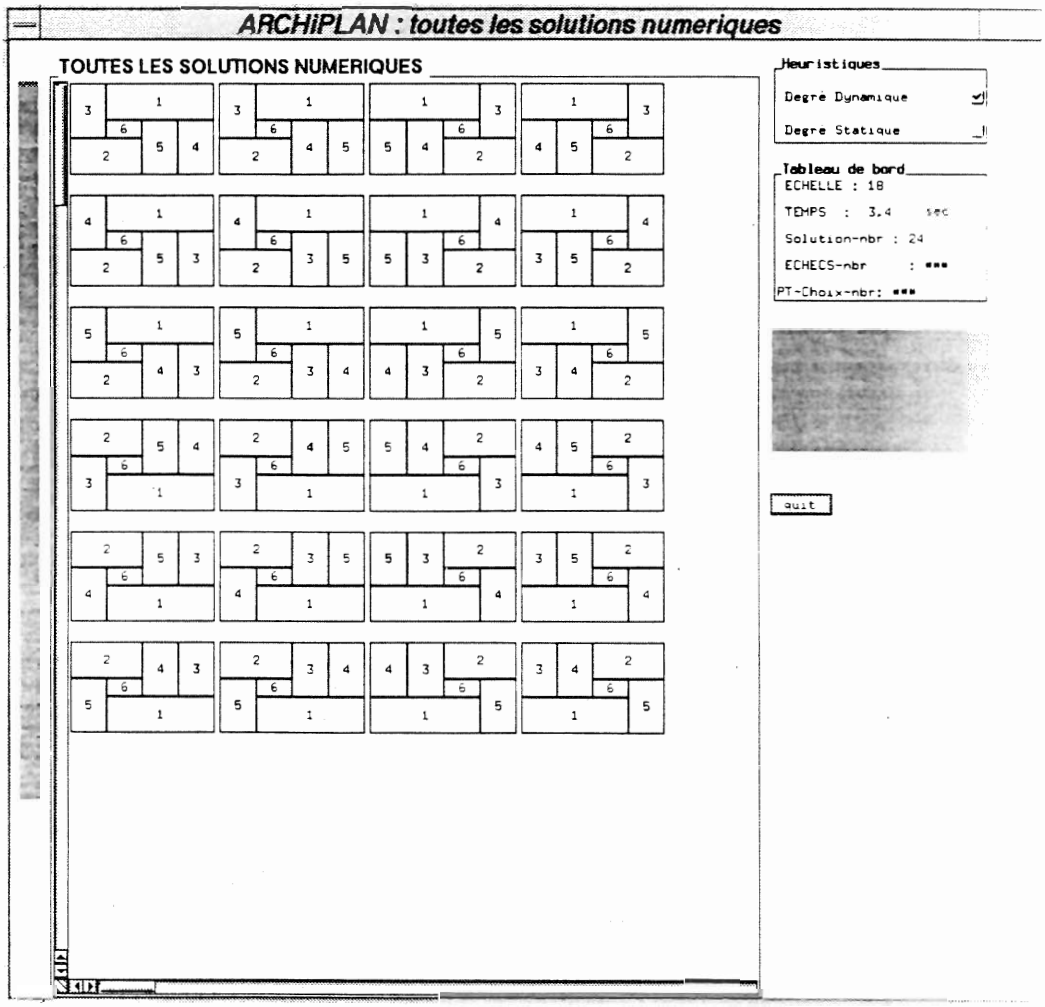


Figure 6.2 : Les 24 solutions du problème de *Pfefferkorn* (avec les solutions symétriques)

La résolution du problème de *Pfefferkorn* (sans les solutions symétriques) s'est effectué en 0,69 secondes (temps CPU) et 3,4 secondes (temps CPU) avec les solutions symétriques, contrairement à EAAS [Charman, 1995] où les temps sont respectivement de 6,84 secondes et 23,76 secondes.

6.1.2. Le problème de Laurière

Il s'agit d'une variante du problème de *Pfefferkorn*. Dans ce problème, les rectangles peuvent tourner (deux orientations possibles 0° et 90°), les autres données du problème reste inchangées.

- **Solutions numériques**

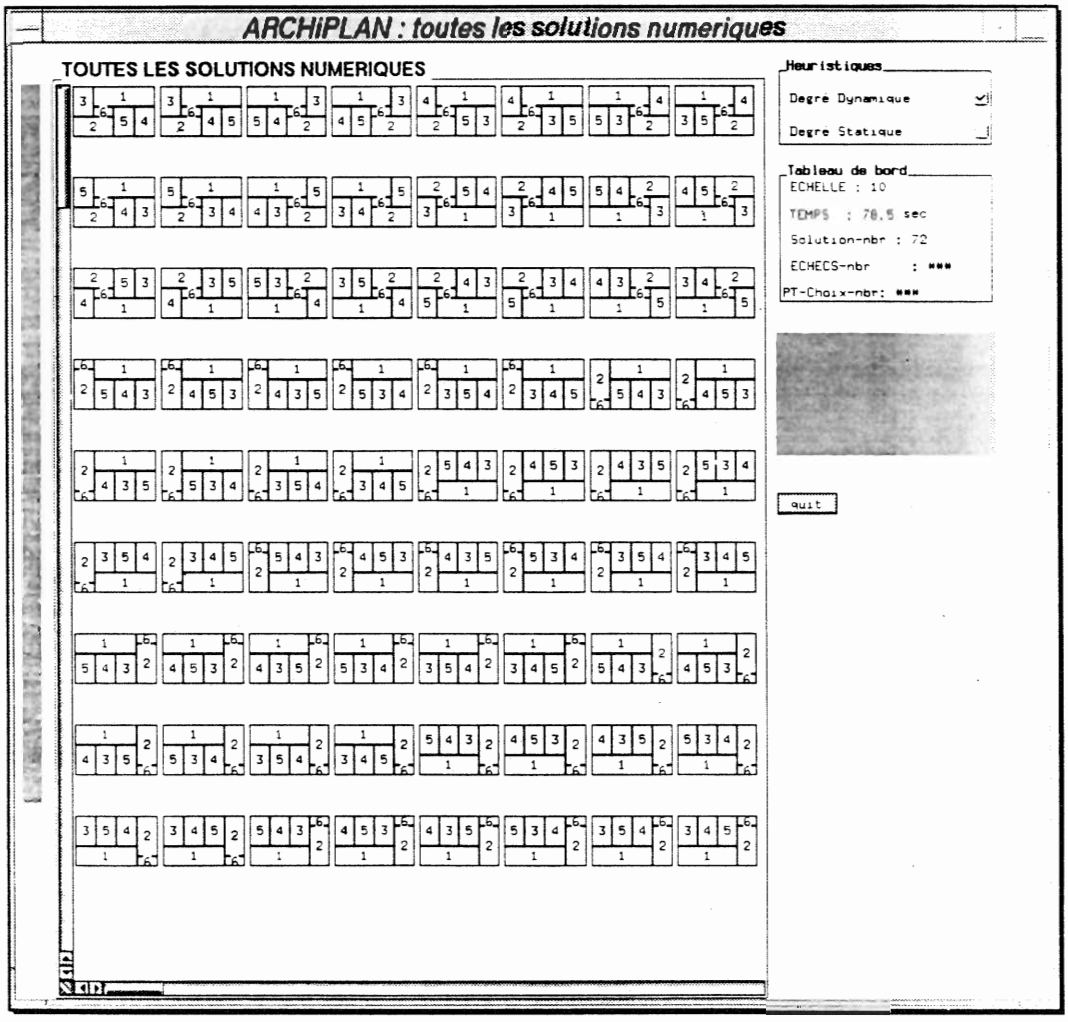


Figure 6.3 : Les 72 solutions numériques du problème de *Laurière* (avec les solutions symétriques)

nous avons résolu le problème de *Laurière* en 78,5 secondes (temps CPU).

6.1.3. Le problème de Tong

Il s'agit de placer dans un rectangle de dimension 8x5 quatre rectangles de dimensions variables entre 4 et 9.

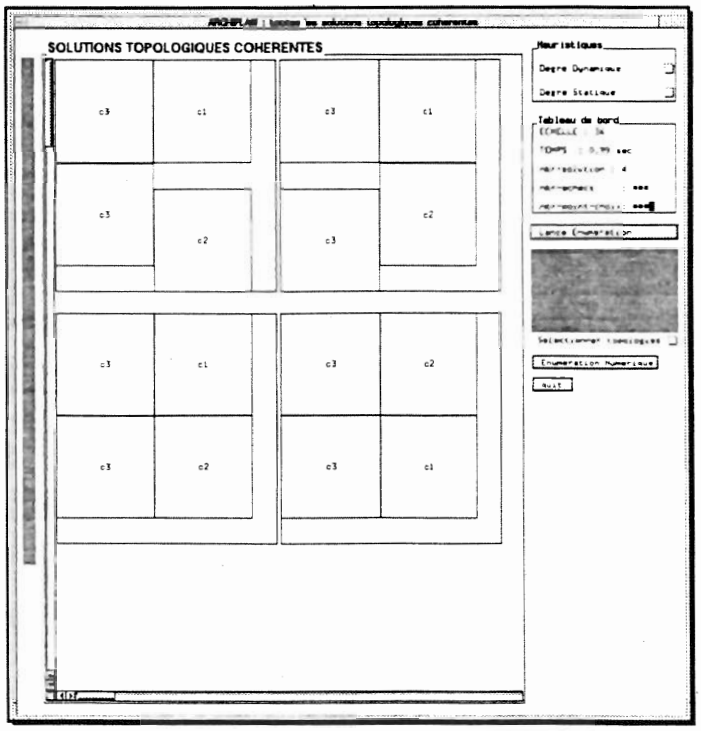


Figure 6.4 : Les quatre solutions topologiques du problème de *Tong*

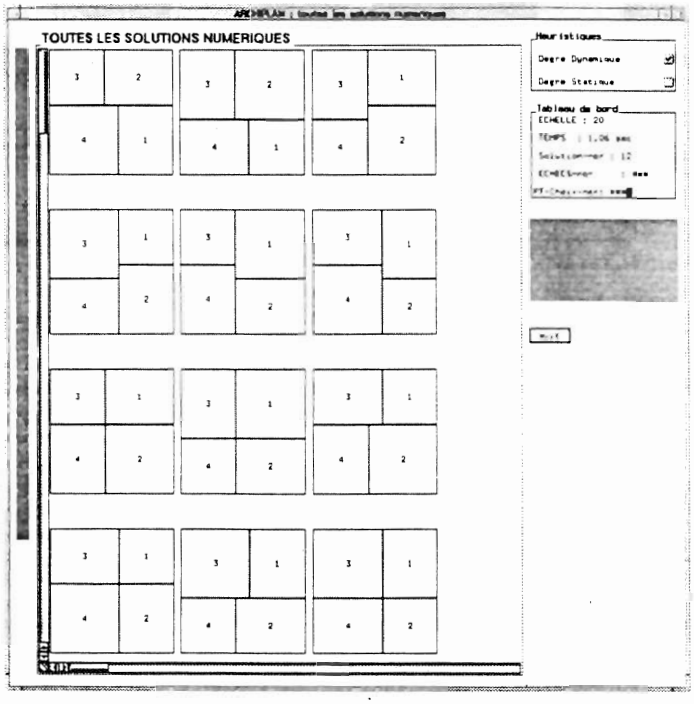


Figure 6.5 : Les 12 solutions numériques excluant les solutions symétriques

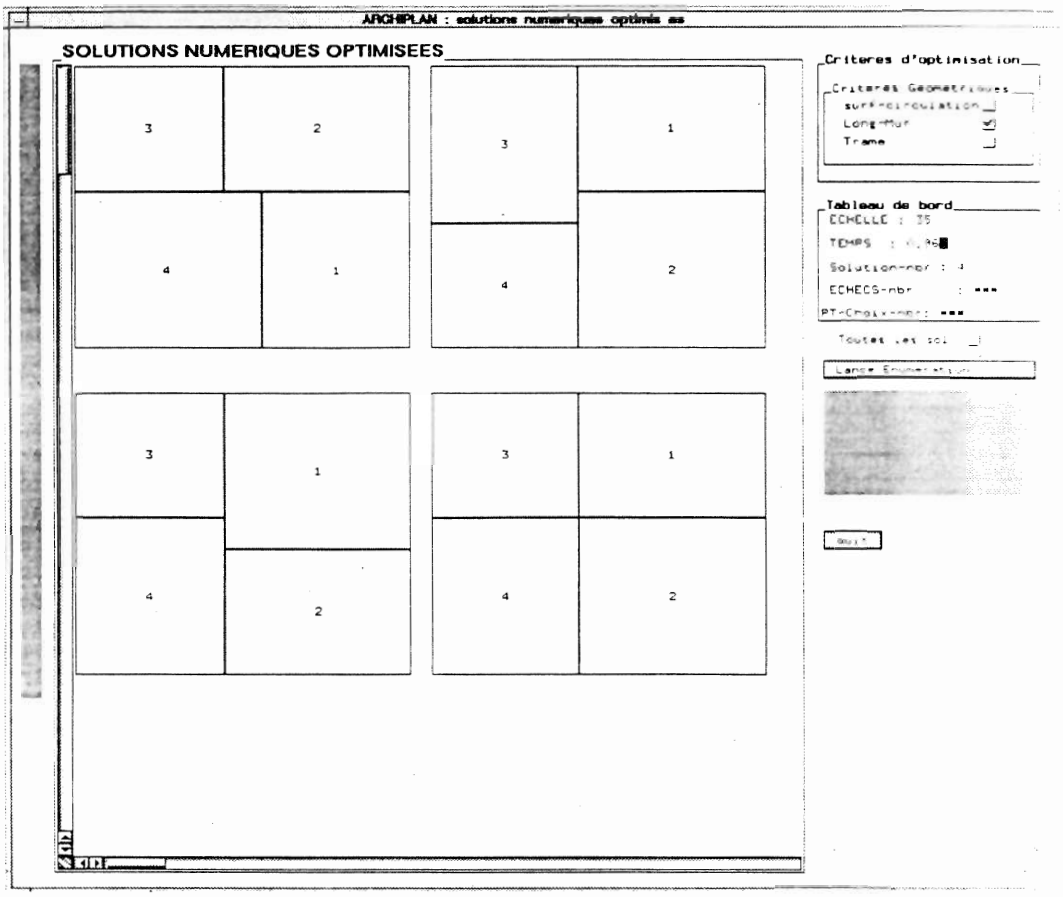


Figure 6.6 : Quatre solutions du problème de *Tong* optimisant le périmètre des rectangles

Nous sommes les seuls à avoir proposé les 4 solutions topologiques et les 4 solutions optimales (en minimisant les périmètres des rectangles) du problème de *Tong*. Nous avons vu que les solutions minimisant les longueurs des murs sont celles qui se rapprochent le plus des espaces carrés (cf. *chapitre 4*). Les 288 solutions numériques ont été obtenues en 11,89 secondes (temps CPU) contrairement aux 236,4 secondes (temps CPU) de EAAS [Charman, 1995].

6.1.4. Le problème de Colmerauer (Col9)

Il s'agit de placer dans un rectangle de dimension 33x32 neuf carrés parfaits dont les côtés sont égaux à 1, 4, 7, 8, 9, 10, 14, 15 et 18.

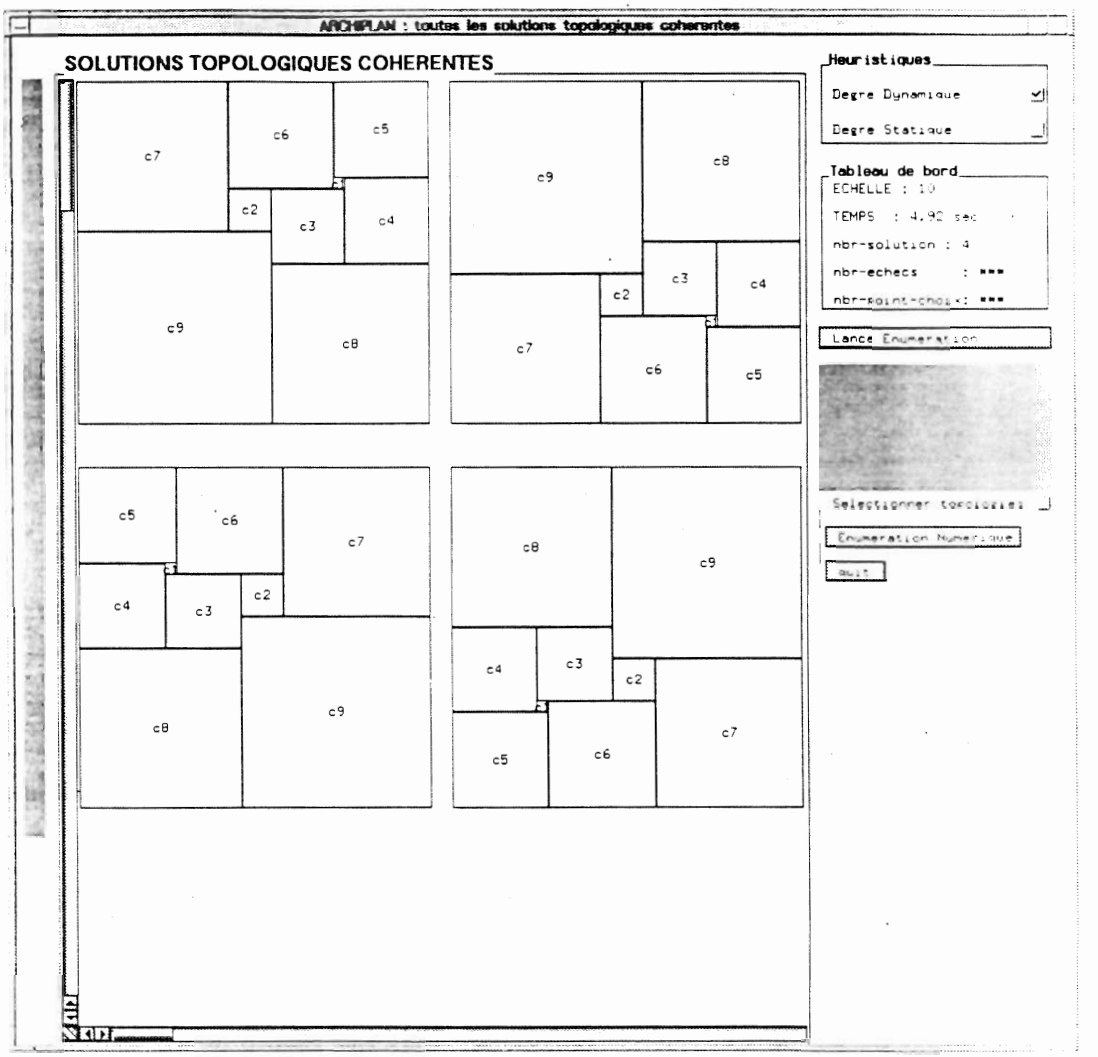


Figure 6.7 : Les quatre solutions des 9 carrés parfaits de *Colmerauer* [Colmerauer, 1990]

Nous avons résolu le problème des 9 carrés parfaits en 4,92 secondes (temps CPU) contrairement à 12,45 secondes dans EAAS [Charman, 1995].

6.1.5. Le problème de Maculet (logement F5 avec deux couloirs)

Il s'agit de concevoir un plan de sol d'un logement F5 de 120 m^2 de surface avec 10 espaces. Cet exemple a été proposé par *Maculet* comme "benchmark" pour des problèmes de placement.

• Cahier des charges et schéma fonctionnel

Nous avons indiqué dans le tableau 6.1 les contraintes dimensionnelles du problème. Un module de distance correspond à 1 mètre.

activite	surface S	L-min	W-min
RDC	[12, 10]	12	10
Séjour	[33 ,42]	4	4
Cuisine	[9 ,15]	3	3
SDB	[6, 9]	2	2
WC	[36, 60]	1	1
couloir2	[1, 12]	1	1

activité	surface S	L-min	W-min
Chambre1	[11, 15]	3	3
Chambre2	[11, 15]	3	3
Chambre3	[11, 15]	3	3
Chambre4	[15, 20]	3	3
couloir1	[1, 12]	1	1

Tableau 6.1 : Contraintes dimensionnelles du problème de *Maculet* [Maculet, 1991]

Les contraintes entre les espaces sont définies par :

- Les contraintes entre les locaux (contraintes binaires ou K-aires)
  - Séjour est *sur-la-façade-sud* ou *sur-la-façade-ouest*,
  - Cuisine est *sur-la-façade-sud* ou *sur-la-façade-nord*,
  - Salle de bains et wc peuvent ne pas être éclairés par la lumière naturelle,
  - Chambre1 est *sur-la-façade-sud* ou *sur-la-façade-nord*,
  - Chambre 2 est *sur-la-façade-sud* ou *sur-la-façade-nord*,
  - Chambre3 est *sur-la-façade-sud* ou *sur-la-façade-nord*,
  - Chambre 4 est *sur-la-façade-sud* (chambre principale),
  - Tous les locaux, hormis la cuisine, sont *adjacents* avec l'un des deux couloirs sur au moins 1 mètre de longueur de contact,
  - Le séjour est *adjacent* à la cuisine,
  - la cuisine est *adjacente* à la salle de bain,
  - Les wc sont *adjacents* à la cuisine ou à la salle de bain,
  - les deux couloirs sont *adjacents*,
  - l'entrée principale est par le séjour.

Remarque : Dans un premier temps nous avons repris ce cahier des charges tel qu'il à été donné par *Maculet*. Dans un second temps, nous avons préféré, pour des raisons

fonctionnelles, poser une adjacence entre la cuisine et l'un des deux couloirs et prévoir une entrée par l'un des deux couloirs. Nous présentons par la suite le résultat du problème de Maculet modifié.

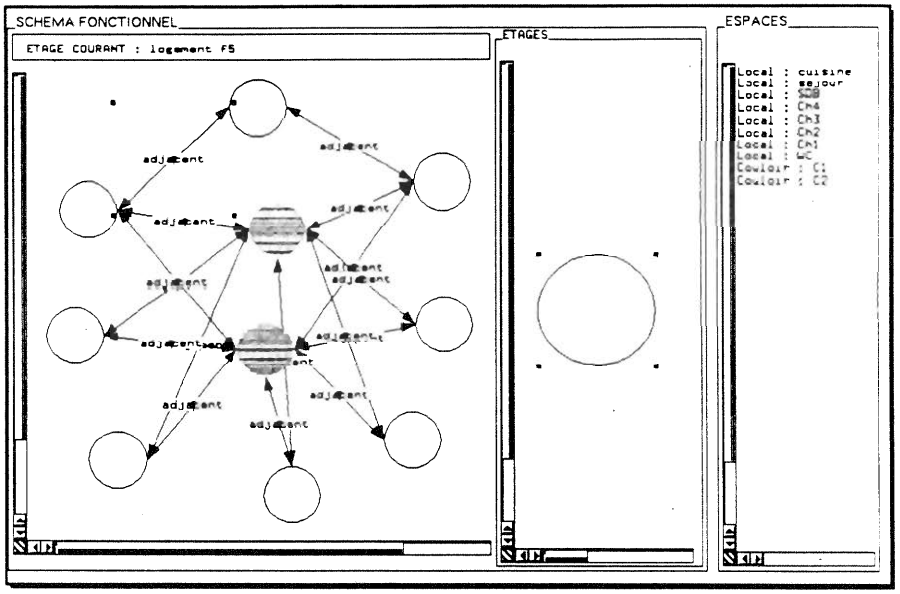


Figure 6.8 : Schéma fonctionnel du logement F5 à deux couloirs

- Quelques solutions topologiques

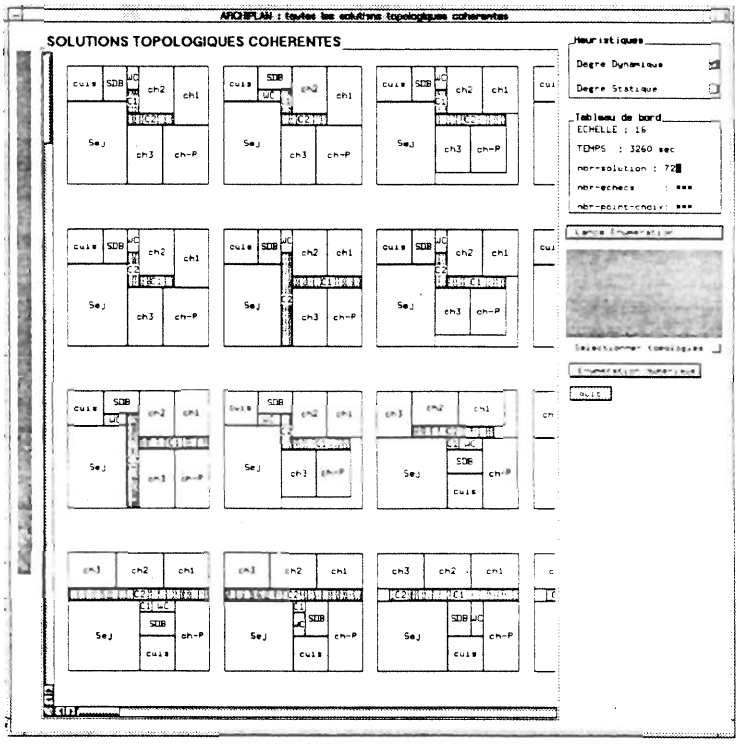


Figure 6.9 : Quelques solutions topologiques (parmi les 72) du problème de *Maculet*

- **Solutions numériques optimisées**

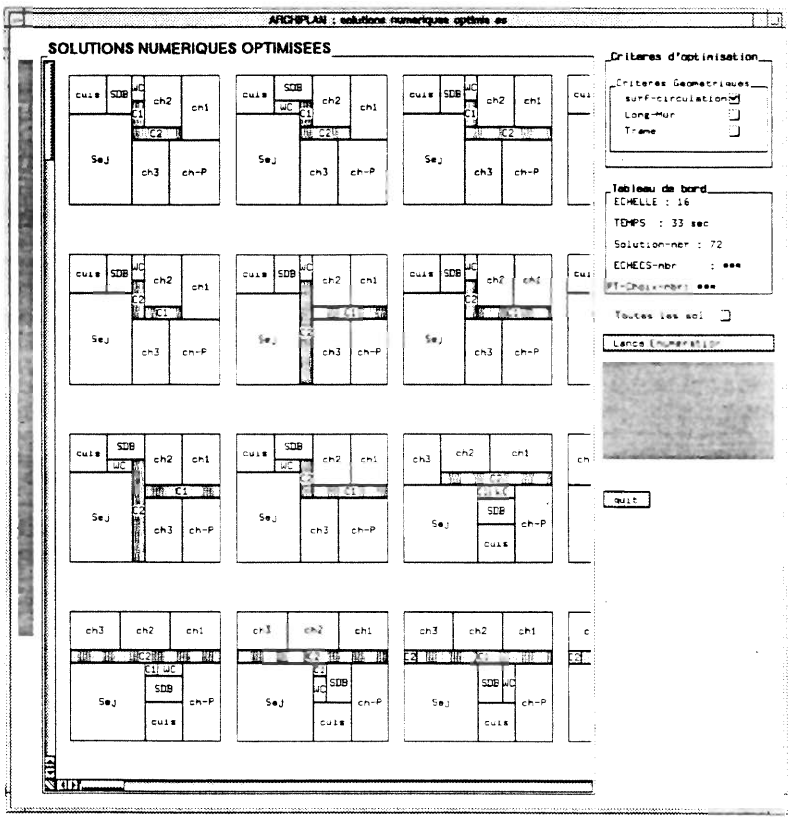


Figure 6.10 : Quelques solutions numériques optimisées (une solution optimale par solution topologique)

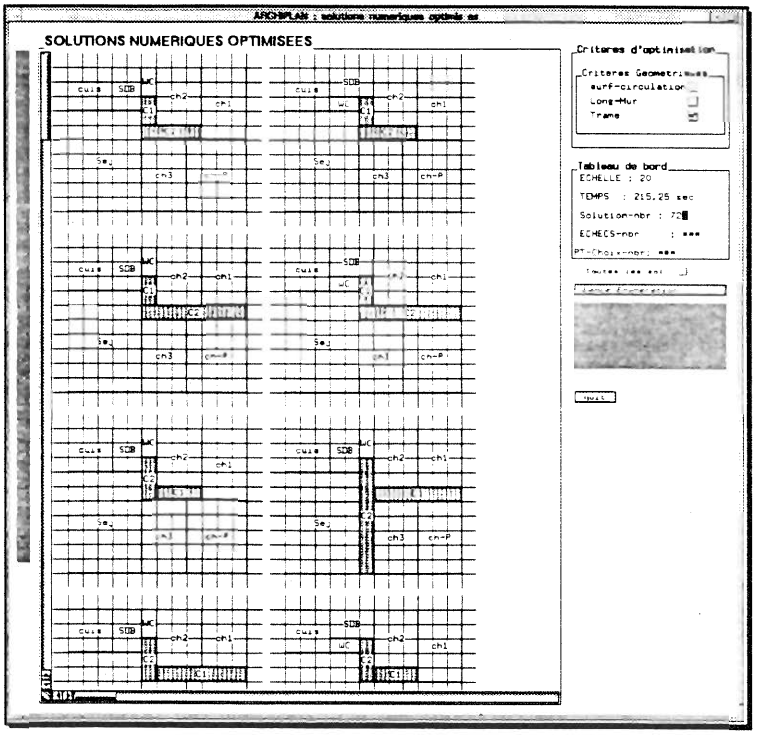


Figure 6.11 : Quelques solutions numériques optimisées de solutions topologiques avec une visualisation d'une trame de 1x1 (une solution optimale par solution topologique)

- **Modification du problème de Maculet** (adjacence entre la cuisine et l'un des deux couloirs et adjacence d'un des deux couloirs avec l'extérieur), ce qui se rapproche beaucoup plus d'un plan type de logement F5 (cf. Figure 6.13).

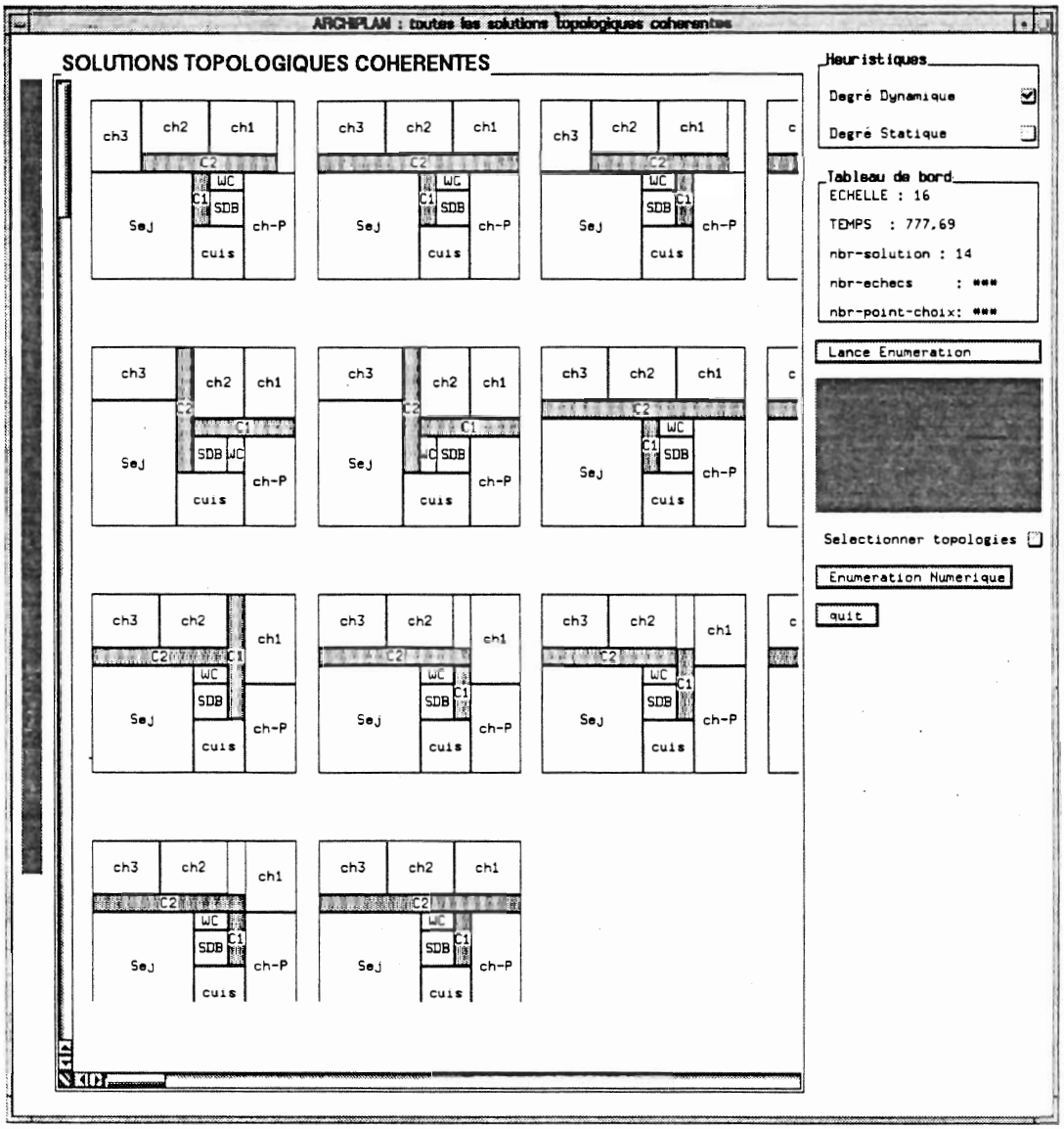


Figure 6.12 : Les 14 solutions topologiques du problème de *Maculet* modifié.

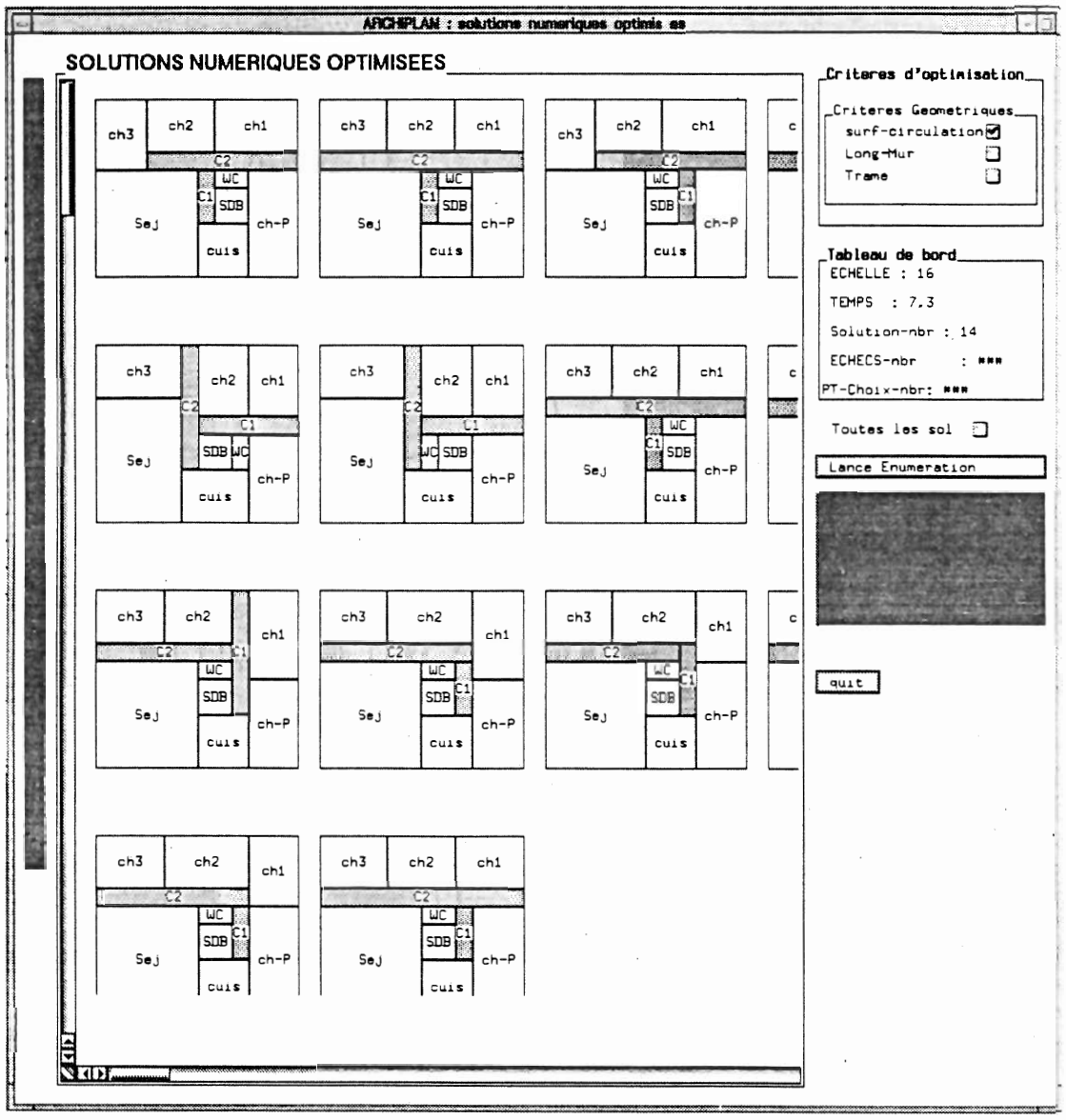


Figure 6.13 : Les 14 solutions numériques optimisées (une solutions optimale par solution topologique) du problème de *Maculet* modifié.

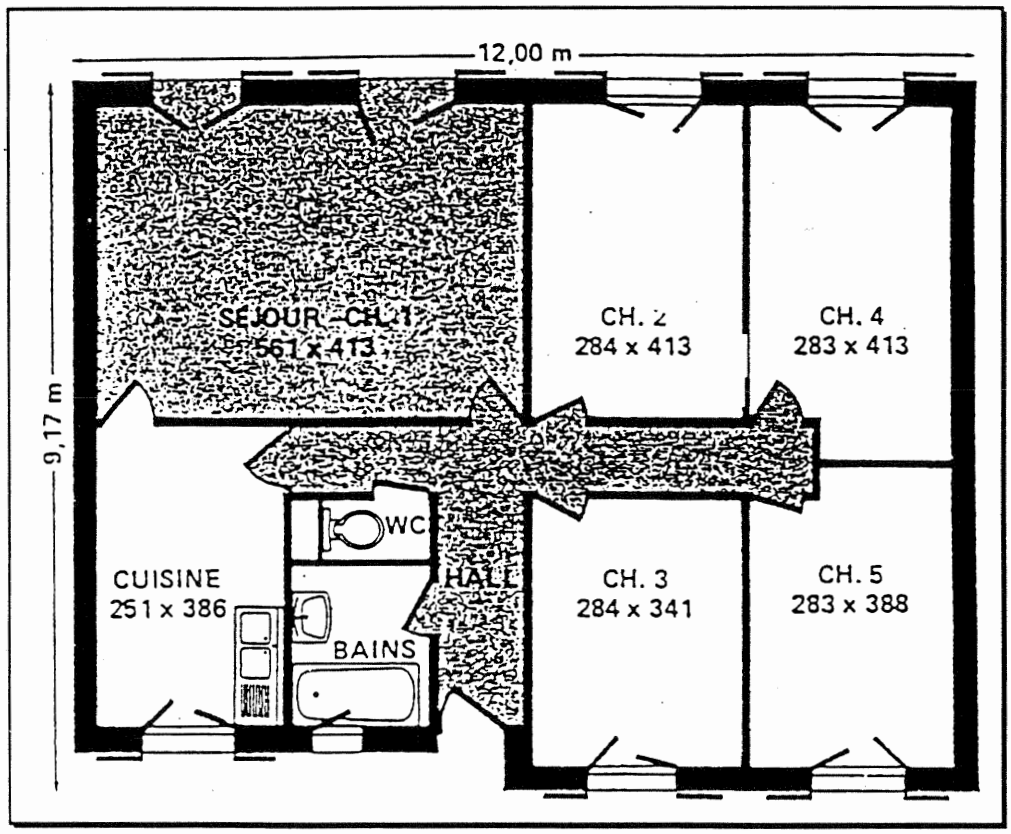


Figure 6.14 : Exemple d'un plan type de logement F5

Nous sommes les seuls à avoir obtenu les solutions topologiques et optimales du problème de *Maculet*. Aucune comparaison n'est donc possible avec d'autres travaux. Nous avons obtenu les 72 solutions topologiques en 3200 secondes (temps CPU) et les 72 solutions optimales en 34 secondes (temps CPU). Le seul niveau de comparaison se fait au niveau de l'obtention de toutes les solutions, *Maculet* a proposé le problème mais ne l'a pas résolu. Par contre *Charman* l'a résolu (solution numériques) en 5400 secondes (temps CPU).

6.2. Nouveaux "benchmarks" proposés

Tous les exemples de cette section sont des propositions de "benchmarks" pour les travaux futurs sur le placement.

6.2.1. Le problème de la maison individuelle (R+1)

Nous traitons dans cet exemple un bâtiment avec deux niveaux (RDC et premier étage). Cet exemple exploite bien la fonctionnalité multi-étages d'ARCHiPLAN.

- **Cahier des charges**

Nous avons indiqué dans le tableau 6.2 les contraintes dimensionnelles du problème. Le module de distance correspond à 50 centimètres.

activité	surface S	L-min	W-min
RDC	[320 ,320]	20	16
Séjour	[72 ,128]	6	6
Cuisine	[36 ,60]	5	5
SDB/WC	[16, 36]	4	4
Bureau	[36, 60]	6	6
Couloir	[4, 64]	3	3
Escalier	[24, 28]	4	6
Couloir2	[4, 64]	3	3

activité	surface S	L-min	W-min
1er étage	[320, 320]		
Chambre1	[48, 60]	6	6
Chambre2	[48, 60]	6	6
Chambre3	[48, 60]	6	6
Chambre4	[48, 72]	6	6
SDB1	[16, 36]	4	4
SDB2	[16, 36]	4	4
Balcon	[12, 24]	3	3

Tableau 6.2 : Contraintes dimensionnelles du (R+1)

Les contraintes entre les espaces sont définies par :

- les contraintes entre les étages :
  - le *1er-étage* est *SUR* le *R.D.C*,
  - l'escalier *Communique-entre* le *1er-étage* et le *R.D.C*,
- les contraintes entre les espaces du *R.D.C* :
  - tous les espaces du *R.D.C* sont *adjacents* avec le couloir sur 1m de longueur de contact, au minimum,
  - le séjour est *sur-la-façade-sud*,
  - la *cuisine* et le *séjour* sont *adjacents* sur 1m de longueur de contact, au minimum,
  - la cuisine est *sur-la-façade-sud* ou *sur-la-façade-nord*,
  - la *cuisine* et la *SDB* sont *adjacents*,
- les contraintes entre les espaces du *1er-étage* :

- tous les espaces du *1er-étage* (à part le balcon) sont *adjacents* avec le *couloir2* sur 1m de longueur de contact au minimum,
- la *chambre4* et la *SDB2* sont *adjacents* avec 1m de longueur de contact au minimum,
- la *chambre4* et le *balcon* sont *adjacents* avec 1m de longueur de contact au minimum,
- le *balcon* est *sur-la-façade-sud*

- **Schéma fonctionnel (organigramme)**

Le cahier des charges est entré dans le logiciel ARCHiPLAN de manière graphique à travers la construction d'un graphe fonctionnel (cf. Figures 6.16 et 6.17).

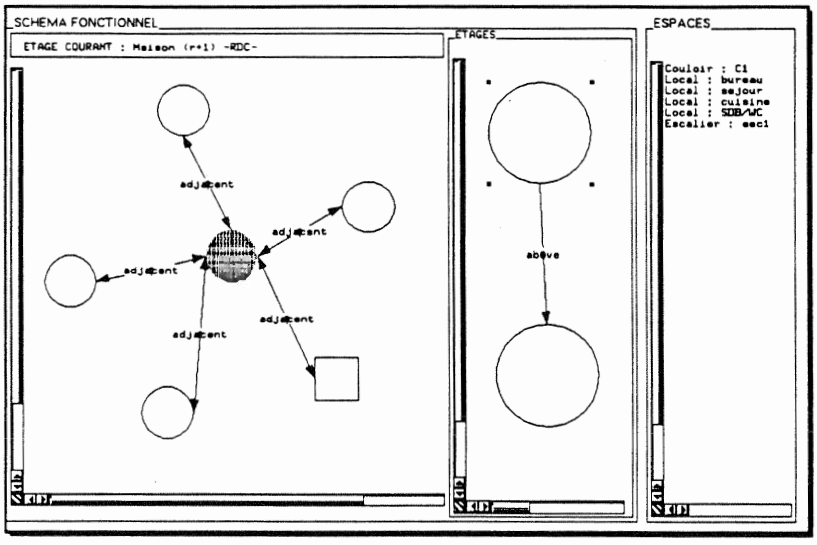


Figure 6.16 : Le schéma fonctionnel de la maison individuelle (RDC)

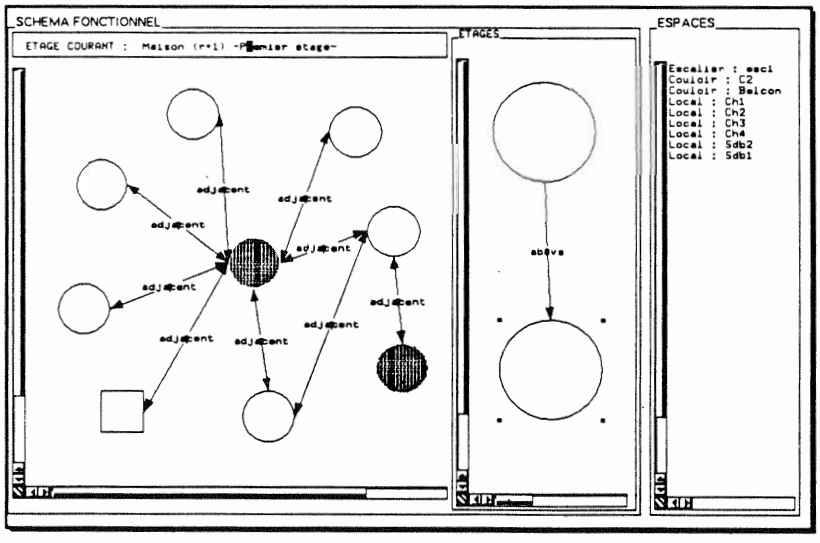


Figure 6.17 : Le schéma fonctionnel de la maison individuelle (premier étage)

- **Quelques solutions topologiques**

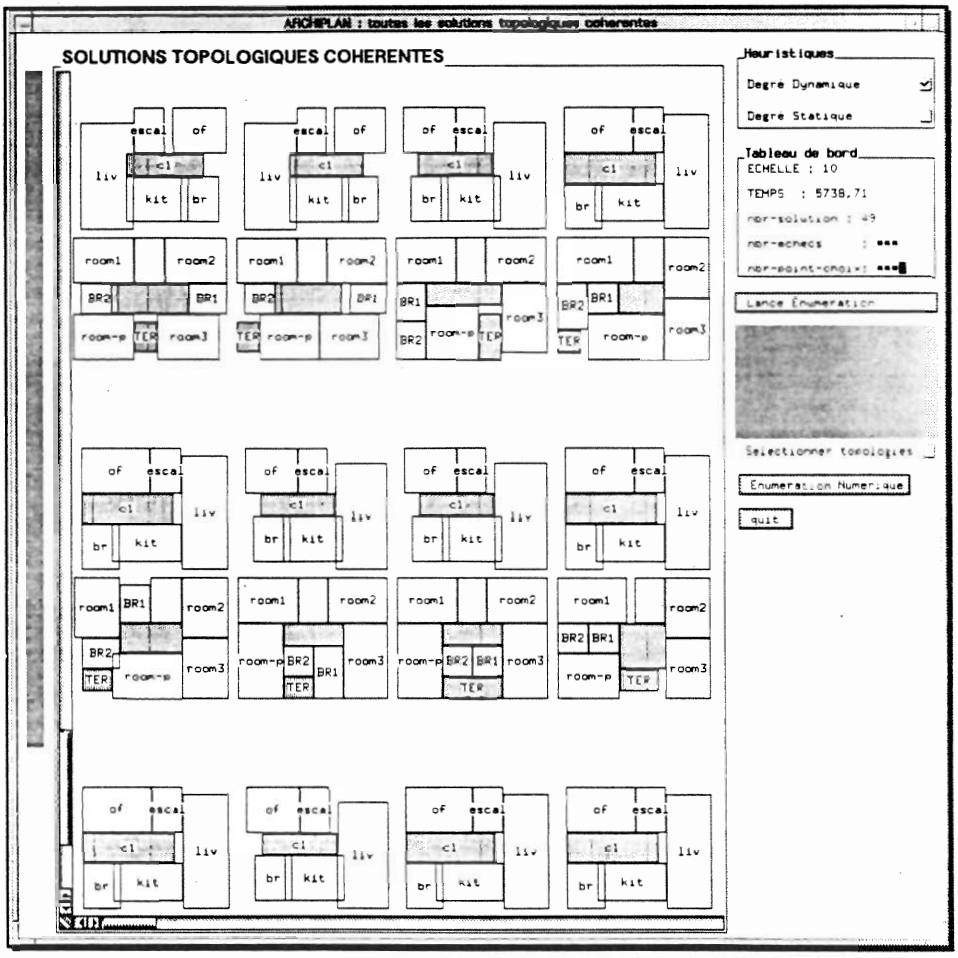


Figure 6.18 : Quelques solutions topologiques (parmi les 49) de la maison individuelle

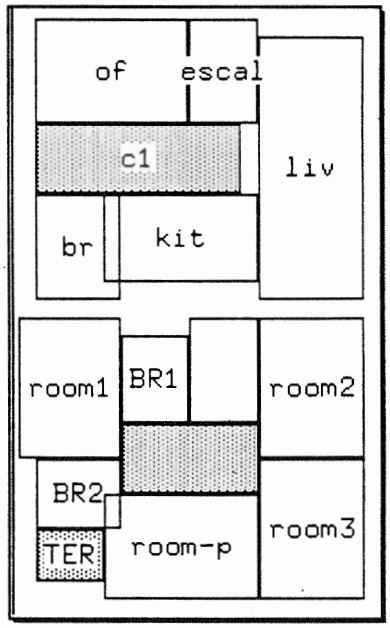


Figure 6.19 : Illustration d'une solution topologique

- **Quelques solutions numériques optimisées**

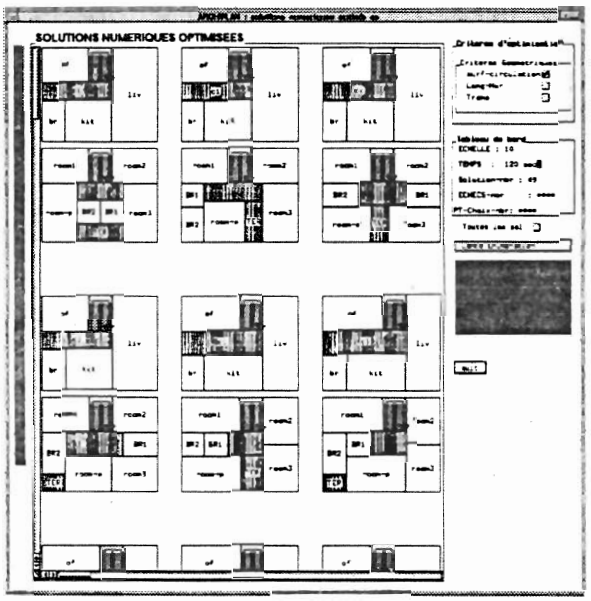


Figure 6.20 : Quelques solutions numériques optimisées (une par solution topologique) de la maison individuelle

Nous avons obtenu les 49 solutions topologiques en 5738 secondes (temps CPU). Les 49 solutions optimales ont été obtenues en 120 secondes (temps CPU).

6.2.2. Le problème de l'ensemble de bureaux

C'est un problème complexe avec 16 espaces. Nous avons présenté cet exemple pour bien montrer la possibilité de pouvoir spécifier des espaces de placement de forme autre que rectangulaire. Dans ce cas, on parlera plutôt de *contour de placement de l'espace de placement* (RDC), qui est l'espace englobant, moins le patio dont la position et les dimensions sont instanciées (cf. le Tableau 6.3).

- **Cahier des charges**

Nous avons indiqué dans le tableau 6.3 les contraintes dimensionnelles du problème. Un module de distance correspond à 1 mètre.

activité	surface S	L-min	W-min
RDC	[120, 120]	12	10
Bureau1	[9, 15]	3	3
Bureau2	[9, 15]	3	3
Bureau3	[9, 15]	3	3
Bureau4	[9, 15]	3	3
Bureau5	[9, 15]	3	3
Bureau6	[9, 15]	3	3
Bureau7	[9, 15]	3	3
Bureau8	[9, 15]	3	3

activité	surface S	L-min	W-min
Bureau9	[9, 15]	3	3
Bureau10	[9, 15]	3	3
toilette1	[6, 9]	2	2
toilette2	[6, 9]	2	2
Reception	[9, 15]	3	3
couloir1	[1, 30]	1	1
couloir2	[1, 30]	1	1
patio	[49, 49]	7	7

Tableau 6.3 : Contraintes dimensionnelles de l'ensemble de bureaux

Les contraintes entre les espaces sont définies par :

- les 10 bureaux et la réception sont *adjacents* avec l'un des deux couloirs sur au moins 1 mètre de longueur de contact,
- la réception est sur l'une des 4 façades extérieures,
- les 2 couloirs sont *adjacents* avec au moins 1 mètre de longueur de contact.

• Quelques solutions topologiques

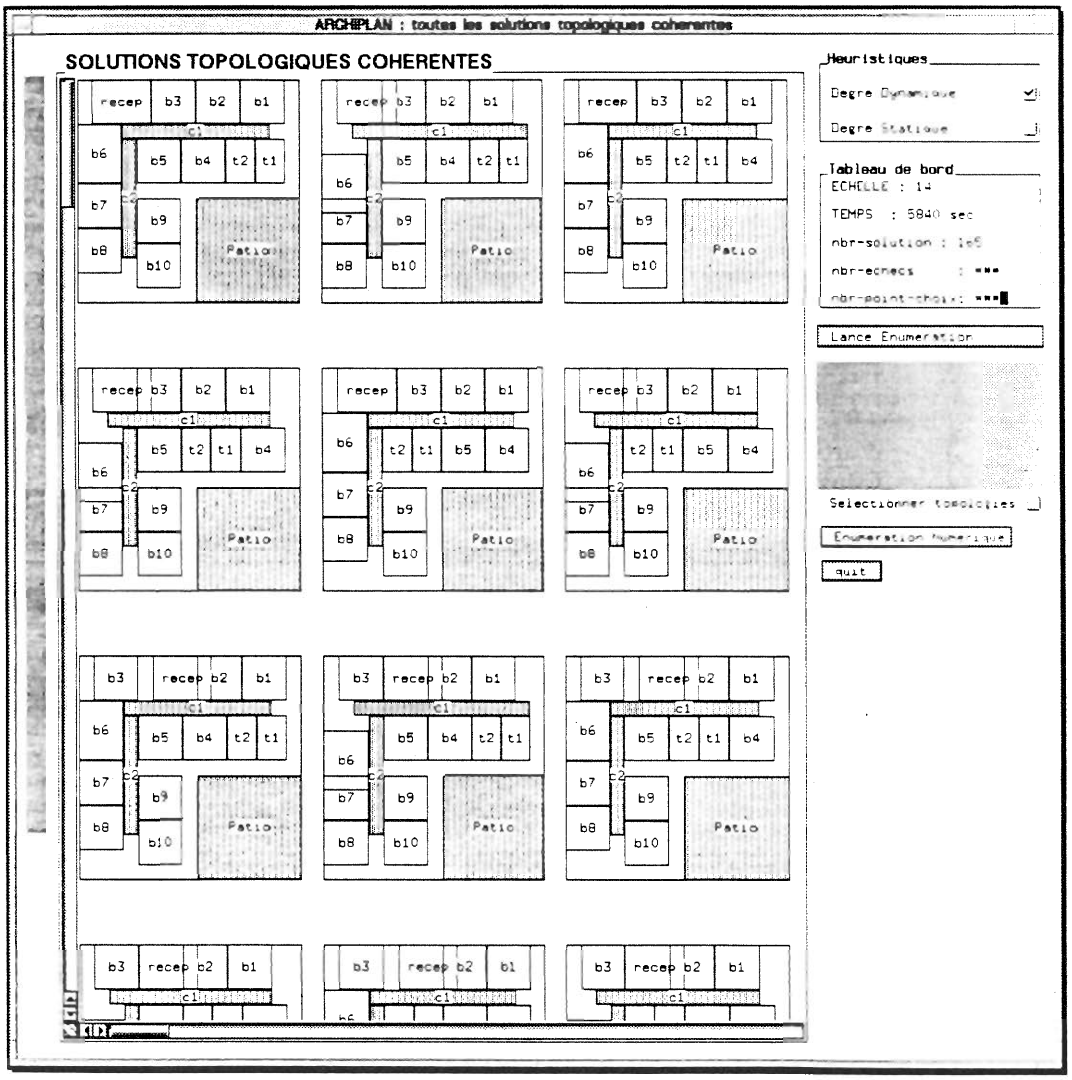


Figure 6.21 : Quelques solutions topologiques (parmi les 102) de *l'ensemble de bureaux*

- **Quelques solutions numériques optimisées**

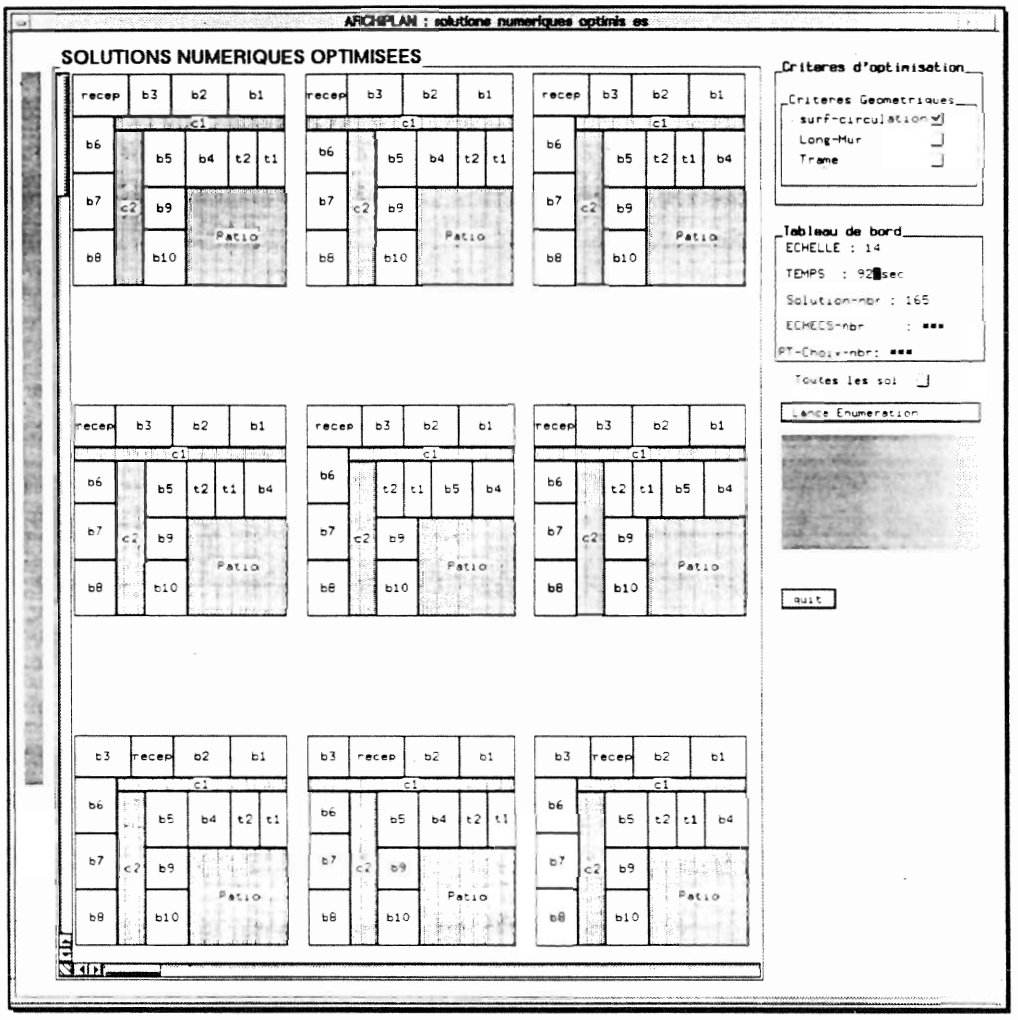


Figure 6.22 : Quelques solutions numériques optimisées de *l'ensemble de bureaux*

Nous avons obtenu les 102 solutions topologiques en 5480 secondes (temps CPU) et les 102 solutions optimales en 92 secondes (temps CPU).

6.2.3. Le problème du jardin d'enfants

Ce problème a été proposé par [Maculet, 1991] mais n'a été jusqu'alors pas résolu.

- **Cahier des charges**

Nous avons indiqué dans le tableau 6.4 les contraintes dimensionnelles du problème. Le module de distance correspond à 1 mètre.

activité	surface S	L-min	W-min
RDC	[1400 , 1400]	28	50
classe1	[90, 100]	6	6
classe2	[90, 100]	6	6
classe3	[90, 100]	6	6
classe4	[90, 100]	6	6
classe5	[90, 100]	6	6
vestiaire1	[10, 15]	2	2
vestiaire2	[10, 15]	2	2
vestiaire3	[10, 15]	2	2
vestiaire4	[10, 15]	2	2
vestiaire5	[10, 15]	2	2
patio	[170, 190]	10	10
préau	[200, 210]	4	4

activité	surface S	L-min	W-min
salle-repas	[140, 160]	10	10
cuisine	[25, 35]	4	4
SDB	[25, 35]	4	4
infirmerie	[10, 15]	3	3
entrée	[25, 35]	4	4
bureau-dir	[10, 15]	3	3
bureau-ins	[10, 15]	3	3
salle-at	[12, 18]	3	3
rangement	[4, 8]	2	2
salle-rithm	[140, 160]	10	10
toillette	[4, 4]	2	2
couloir	[1, 100]	1	1

Tableau 6.4 : Contraintes dimensionnelles du jardin d'enfants

Les contraintes entre les espaces sont définies par :

- les 5 classes, le patio, la salle de repas, la SBD et la salle d'attente sont *adjacentes* avec le préau sur au moins 1 mètre de longueur de contact,
- la salle d'attente, le bureau des institutrices, le bureau de la directrice, la salle de rangement, les toilettes et l'entrée sont *adjacentes* avec le couloir sur au moins 1 mètre de longueur de contact,
- toutes les classes sont sur l'une des façades extérieures,
- les 5 classes ont la même surface (contrainte de *ratio*),
- chaque vestiaire est *adjacent* à une classe,
- la salle de rythme est *adjacente* au patio avec au moins 1 mètre de longueur de contact,
- la salle de rythme est *adjacente* avec l'infirmerie avec une distance maximale de 4 m,
- la salle d'attente est *adjacente* avec l'entrée sur au moins 1 mètre de longueur de contact.

Nous avons considéré la symétrie entre les 5 classes étant données les contraintes identiques portant dessus. Nous avons inclu chaque vestiaire dans une classe.

- **Quelques solutions topologiques**

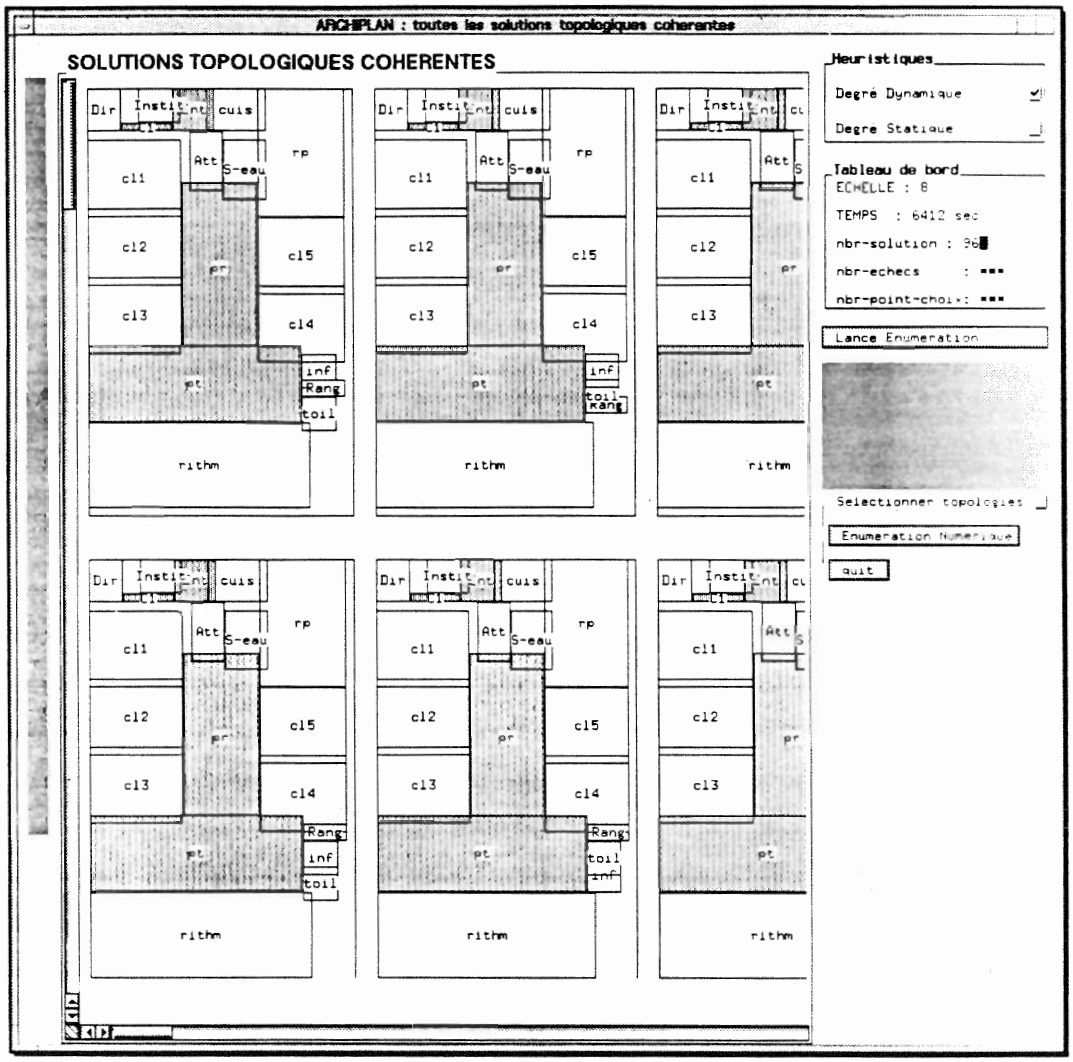


Figure 6.23 : Quelques solutions topologiques (parmi les 96) du jardin d'enfants

- **Quelques solutions numériques optimisées**

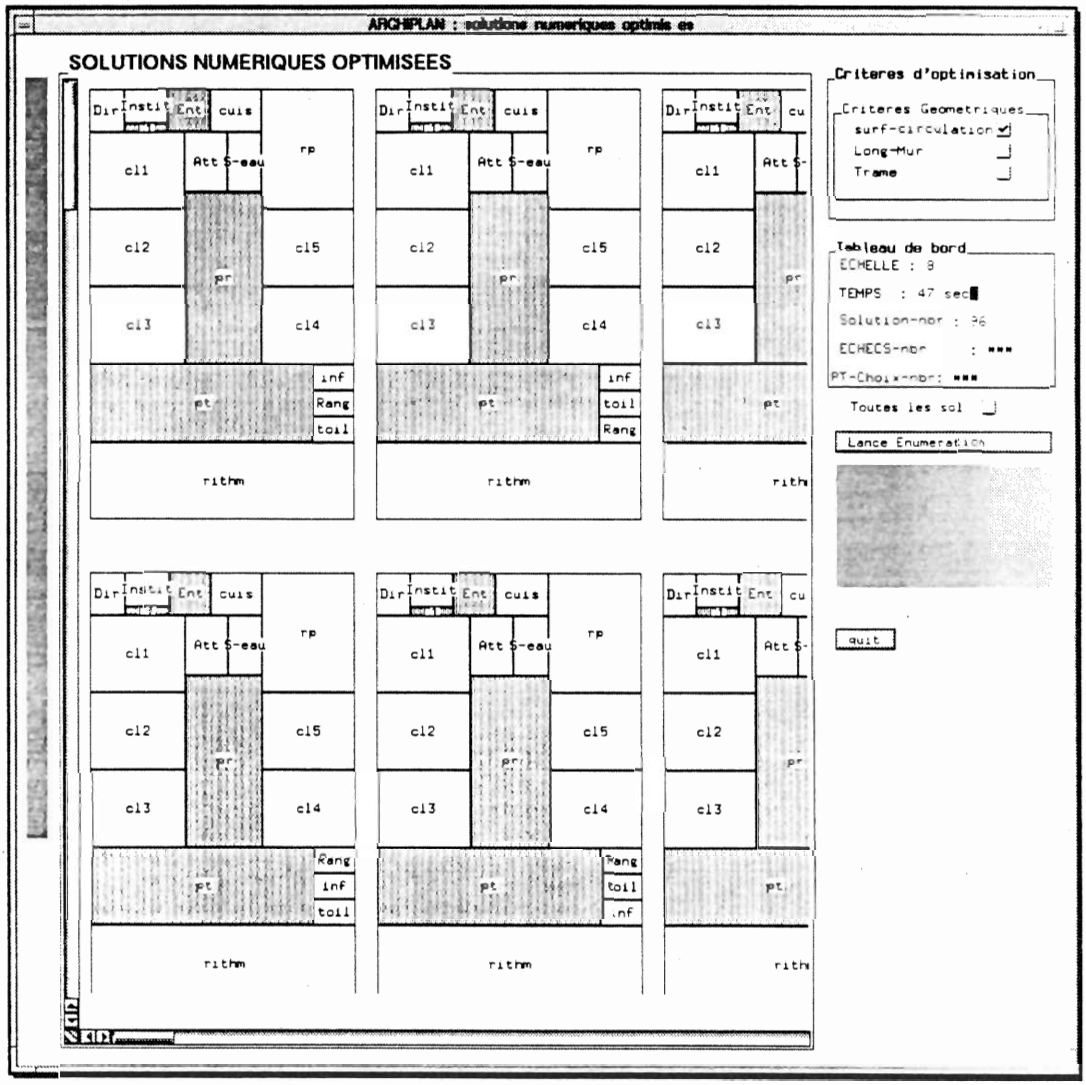


Figure 6.24 : Quelques solutions numériques optimisées (parmi les 96) du jardin d'enfants

Nous avons obtenu les 96 solutions topologiques en 6480 secondes (temps CPU) et les 96 solutions optimales en 47 secondes (temps CPU).

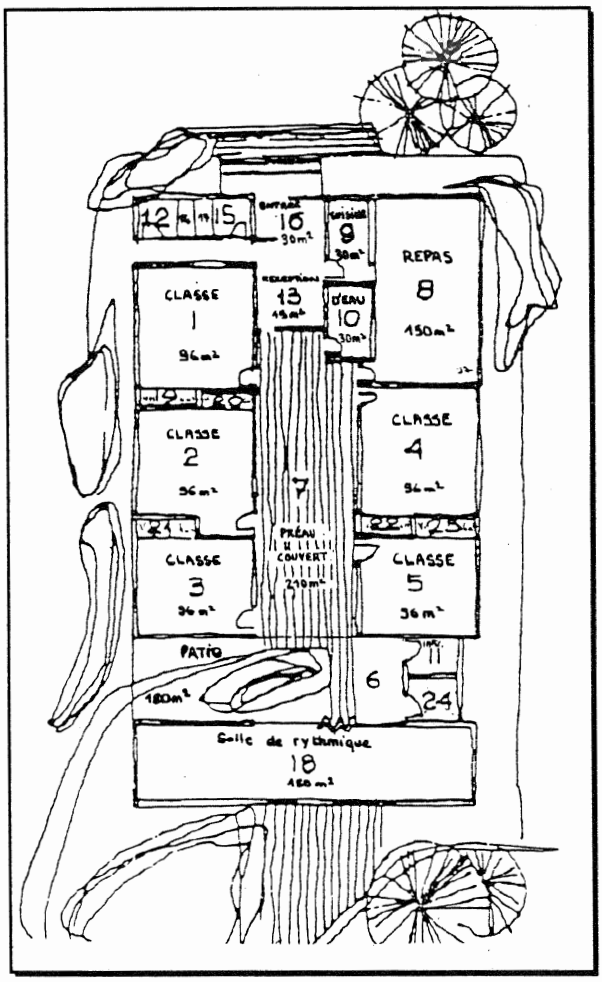


Figure 6.21 : Illustration d'un plan de jardin d'enfants ressemblant au solutions obtenues, trouvé dans [Maculet, 1991].

6.3. Conclusion

Nous avons, dans ce chapitre, validé notre approche en comparant nos travaux aux travaux précédents et également, en proposant de nouveaux "benchmarks" pour les travaux futurs.

Nous avons démontré l'efficacité et la faisabilité d'ARCHiPLAN au niveau de la résolution de problèmes de placement de taille moyenne (32 espaces) pour des problèmes réels de conception en architecture.

Chapitre 7

Conclusion et perspectives

7.1. Notre contribution

Nous avons présenté une nouvelle méthodologie de conception fonctionnelle en architecture qui préfigure une nouvelle génération de logiciels de CAO. Notre apport dans cette thèse se situe :

• **dans son modèle à trois niveaux** : ***fonctionnel, topologique et numérique,*** qui, non seulement évite le problème de l'explosion combinatoire des solutions numériques, mais également se conforme à la méthodologie de conception des architectes. En effet, il a été montré que la représentation graphique d'une solution topologique pourrait être équivalente à une *esquisse*. D'autre part, nous avons exploité le fait qu'un architecte raisonne, en conception préliminaire, en termes d'*esquisses*. Pour une *esquisse* donnée, il aimerait savoir quelle est la *coordination dimensionnelle* optimale pour des critères aisément explicitables. Par contre, le choix entre une esquisse et une autre, qui fait intervenir des considérations plus subjectives (*aspects artistiques*), reste de son ressort.

• **dans l'enrichissement du modèle de connaissances architecturales** pour des problèmes d'allocation spatiale et son extension aux **circulations verticales**, ce qui nous a permis de tenir compte des cahiers des charges de bâtiments à **plusieurs étages**.

• **dans la recherche de solutions numériques optimales**. ARCHiPLAN permet, en effet, de ne retenir que les meilleures solutions satisfaisant des critères architecturaux (minimiser les surfaces de circulation, minimiser la longueur des murs, de se rapprocher d'une solution tramée, etc).

• **dans l'introduction d'une nouvelle heuristique dynamique** basée sur le « *contour extérieur courant* ». Cette heuristique a permis d'augmenter l'efficacité de l'algorithme de recherche des solutions topologiques.

• **dans l'enrichissement du modèle de contraintes** :

  • Nous avons résolu entièrement le problème des solutions redondantes pour chaque orientation des espaces.

  • Au niveau des contraintes fonctionnelles, nous avons proposé la contrainte d'**adjacence généralisée** en introduisant les notions de ***périmètre de protection*** et de ***longueur de contact***.

- Au niveau des contraintes de réduction de l'espace de recherche :
  - nous avons entièrement résolu le problème de **symétrie** entre les espaces, en généralisant la contrainte à toute les orientations.
  - nous avons introduit la contrainte de **réduction des topologies**.

• **dans l'aspect modulaire d'ARCHiPLAN**, aussi bien dû à la programmation orientée objets qu'à la programmation par contraintes (découplage des contraintes et des algorithmes) fait que le cœur d'ARCHiPLAN restera inchangé en cas d'extension de la base des espaces architecturaux, de la base des contraintes ou de la base des critères d'optimisation.

7.2. Comparaison à d'autres travaux

Les travaux les plus proches d'ARCHiPLAN et qui sont basés sur les contraintes sont ceux portant sur EAAS [Charman, 1995], ARCHIPEL [Maculet, 1991] et IDIOM [Smith, 1996].

EAAS comme Archipel présentent des inconvénients majeurs :

- l'énumération des solutions se fait directement au niveau numérique, ce qui nous l'avons vu, est un contresens à double titre : d'une part il y a explosion combinatoire pour un problème peu contraint et, d'autre part, on ne tient pas compte de la méthodologie de conception en architecture où la première évaluation d'une alternative de conception se fait sur une esquisse.
- les solutions numériques sont énumérées sans tenir compte d'aucun critère d'optimisation. Ces deux systèmes énumèrent en vrac les solutions. Dans notre cas, ARCHiPLAN tient compte de certain critères d'architecture afin d'obtenir les meilleures solutions.
- ces approches se limitent à des problèmes à un niveau (R.d.C).

En ce qui concerne IDIOM :

- Alors qu'ARCHiPLAN est un système de synthèse de solutions de placement à partir de contraintes explicites, IDIOM est un système de conception basé sur la capitalisation du savoir architectural (base de cas) difficilement formalisable. Pour une conception donnée, IDIOM identifie les cas de conceptions antérieures proches et les adaptent au nouveau problème par résolution de contraintes.
- IDIOM présente deux niveaux : le concepteur définit lui-même une topologie, contrairement à ARCHiPLAN où les solutions topologiques sont le résultat de

son premier niveau de conception. Le deuxième niveau consiste en une recherche d'un cas similaire dans la base de cas et en son adaptation. IDIOM se situe en aval du processus de conception par rapport à ARCHiPLAN.

- Nous pensons qu'il peut y avoir une complémentarité entre IDIOM et ARCHiPLAN. Si l'architecte ne connaît pas exactement la topologie du bâtiment qu'il désire, ou encore, si en utilisant IDIOM aucune solution n'est trouvée. Il serait possible d'utiliser ARCHiPLAN pour trouver la solution topologique qui intéresse le concepteur. Dans un second temps on pourrait utiliser IDIOM pour trouver la solution numérique la mieux adaptée. De cette façon on utiliserait ainsi les deux systèmes au niveau de leurs points forts.

7.3. Perpectives scientifiques

Outre la perspective d'extension d'ARCHiPLAN à l'implantation industrielle à travers une nouvelle thèse en architecture (cf. §7.3.8), nous avons identifié plusieurs extensions possibles.

7.3.1. Généralisation de l'espace de placement à une forme plus complexe

Nous avons montré dans l'exemple de l'ensemble de bureaux (cf. *chapitre 6*) la possibilité d'utiliser, plutôt que des espaces de placement rectangulaires, des *contours de placement* composés par un assemblage de rectangles : cela correspond à une demande des architectes. La représentation d'un *contour de placement* pourra faire l'objet d'une interface conviviale pour laquelle le concepteur définirait graphiquement le contour. à partir de segments horizontaux et verticaux (cf. Figure 7.1). Il ne resterait alors plus qu'à mettre en œuvre un algorithme de détermination du nombre minimal de rectangles (ce qui est classique) qui, par soustraction à l'espace de placement (englobant) donneraient le *contour de placement*.. Pour ce *contour de placement*, l'espace de placement ainsi que les rectangles déterminés auparavant, seraient automatiquement instanciés. L'algorithme doit privilégier, à nombre de rectangles minimal, les rectangles les plus grands, quitte à ce qu'ils se chevauchent (cf. Figure 7.1) car en condamnant plusieurs fois les mêmes zones, les contraintes risquent de se propager plus vite. Il faut, de plus, modifier quelque peu la contrainte d'adjacence avec le *contour de placement*. On constate que tout ceci ne consiste donc qu'en une « sur-couche » du système informatique actuel.

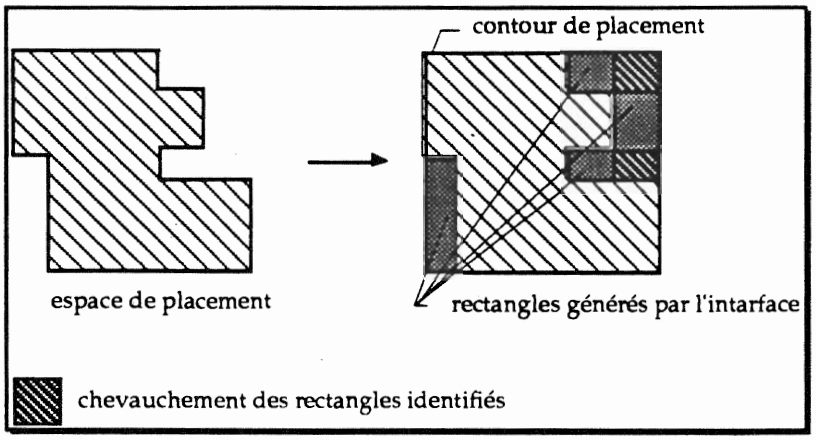


Figure 7.1 : Saisie graphique d'un contour de placement et identification automatique du nombre minimal de rectangles à enlever

7.3.2. Caractérisation plus fonctionnelle des circulations

Il faudrait dans le schéma fonctionnel considérer une circulation sans présupposer de sa forme. En effet, un couloir peut être composé de deux parties (en L, en T, etc), voire de trois parties. Un objet circulation générerait alors un point de choix supplémentaire où chaque choix correspondrait à un nombre de rectangles composant la circulation (un, deux ou trois).

Cette approche peut être utilisée pour avoir des *pièces*. en *L*, Le gros inconvénient est l'augmentation de la complexité du problème. Des contraintes spécifiques pour résorber une partie de la complexité doivent être mises au points.

7.3.3. Enrichissement des critères d'optimisation

Il est possible d'améliorer la phase d'optimisation à deux niveaux :

- On peut étendre la base de critères d'ARCHiPLAN aux :
  - critère de trames respectant la descente de charge ; c'est-à-dire dans les cas où les trames sont différentes d'un étage à l'autre, il faut tout de même avoir une superposition des trames au niveau des descentes de charge.
  - critères acoustiques et thermiques. Ces critères doivent influer sur la position ou la configuration géométrique des pièces (facteur de forme, etc).
- L'optimisation doit être multi-critères avec des pondérations traduisant l'importance relative des critères. Par exemple, nous devrions pouvoir avoir une solution tramé qui minimise les surfaces de circulation.

7.3.4. Contrainte d'adjacence référençant les côtés d'espaces

Nous avons, dans ARCHiPLAN, présenté la contrainte d'*adjacence généralisée*. Celle-ci est basée sur la notion d'orientation géographique (E, O, N, S). Elle ne permet pas du tout de faire référence au côté (cloison) spécifique d'un espace (cf. Figure 7.2).

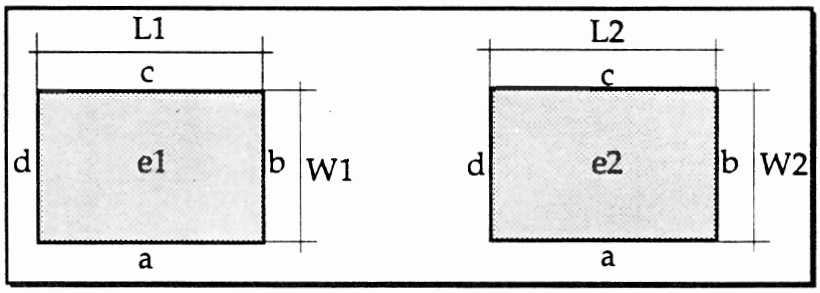


Figure 7.2 : Référence des côtés *a*, *b*, *c* et *d* des espaces (*e1* et *e2*)

Par exemple, pour l'instant on ne peut pas prendre en compte les contraintes suivantes :

- J'aimerais que l'espace *e1* soit adjacent au côté *a* de l'espace *e2*.
- J'aimerais que l'espace *e1* soit adjacent à un côtés relatifs à *L1* (*a* ou *c*) de l'espace *e2*.
- J'aimerais que le côté b de l'espace e1 soit adjacent au côté a de l'espace e2 (quelque soit l'orientation de l'ensemble).
- J'aimerais que l'espace *e2* et l'espace *e3* soient adjacents à un même côté de l'espace *e1* (*a*, *b*, *c* ou *d*).

Pouvoir référencer des côtés ou un des côtés, modifie notre modèle de contraintes (contrainte d'adjacence généralisée) mais ces nouvelles contraintes ont un sens en architecture. Il faudrait essayer de recenser toutes les combinaisons de contraintes possibles et essayer d'en faire un nouveau modèle de contraintes homogène, une sorte de nouvelle algèbre.

7.3.5. Contraintes de flux

La prise en compte des contraintes de flux (personnes comme matières) nous semble un premier pas pour résoudre des problèmes industriels mais peut déjà s'avérer utile dans des projets non industriels comme des immeubles de bureaux ou des centres commerciaux, etc. Par exemple, minimiser tous les parcours vers les sorties de secours (selon les normes en vigueur), revient à **minimiser** des flux, ou encore le flux de l'approvisionnement d'un restaurant **ne doit pas s'intersecter** avec celui des ordures.

7.3.6. Prise en compte de l'incertitude des contraintes fonctionnelles

La prise en compte de l'incertitude sur la pertinence ou la persistance dans le temps des contraintes fonctionnelles semble être une préoccupation majeure de certains concepteur[1]. Nous avons considéré deux types d'incertitudes.

La première est l'incertitude sur la légitimité, l'existence même de la contrainte. En effet, un cahier des charges peut dans certain cas continuer à évoluer en cours de projet. On aura donc, à un moment donné, un *degré de certitude* compris entre 0 et 1, pour chaque contrainte fonctionnelle. Les contraintes fonctionnelles qui ne pourront pas être remises en cause (*degré de certitude*=1) seront considérées comme des contraintes « dures » c'est-à-dire inviolables ; elles seront donc soumises à la propagation de contraintes. Les autres contraintes dites « incertaines » *degré de certitude* compris entre 0 et 1 pourront être violées, mais elles contribueront alors à augmenter le coût de la solution en proportion du degré de viol d'une part et du *degré de certitude* d'autre part.

De même, que la contrainte soit « dure » ou « incertaine », on peut penser modéliser l'incertitude quant au degré de satisfaction des spécifications en proposant la notion de variable contrainte floue. En Figure 7.3, on représente le degré de satisfaction du paramètre longueur de contact d'une contrainte d'adjacence « dure ». Comme pour un nombre flou, le degré de satisfaction μ(d1) culmine à la valeur 1. En Figure 7.4, on représente le degré de satisfaction (ou plutôt d'insatisfaction) d'une contrainte « incertaine » : S<20, dont on ne veut pas se priver totalement des solutions pour S≥20.

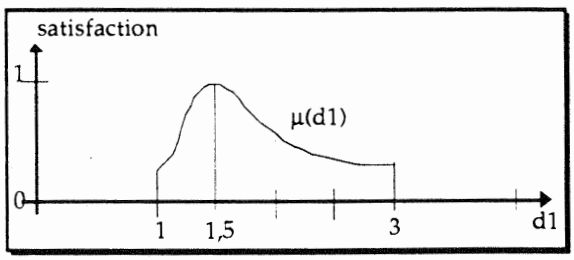


Figure 7.3 : Degré de satisfaction du paramètre *longueur de contact d1*`

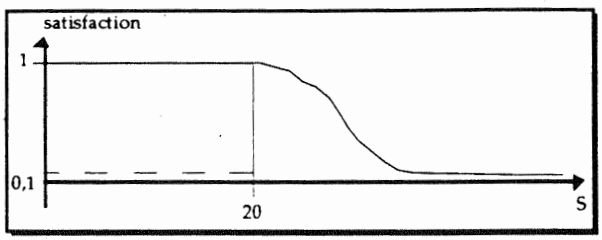


Figure 7.4 : Degré de satisfaction de la contrainte S<20

1. En effet un cahier des charges peut dans certains cas continuer à évoluer en cours de projet.

On peut même, par le même principe, modéliser la préférence de l'orientation relative de 2 espaces. L'adjacence de 2 espaces fait intervenir le point de choix à 4 valeurs {E, O, N, S} dont la Figure 7.5 représente la préférence.

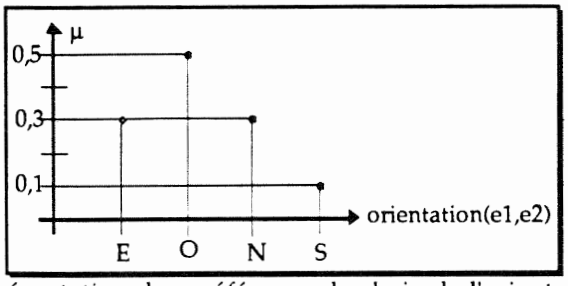


Figure 7.5 : Représentation des préférence de choix de l'orientation relative entre deux espaces

On propose de tenir compte à la fois du degré de certitude de la légitimité de la contrainte et de la courbe de satisfaction des paramètres relatifs à cette contrainte en considérant des coûts, dans la fonction d'optimisation donnés par la formule (x est la valeur courante de la variable contrainte x) :

$$Coût(x)=Degré\text{-}certitude \times (1/\mu(x) - 1)$$

7.3.7. Extension d'ARCHiPLAN à des fonctionnalités pour la réhabilitation

Le principe consiste, à partir d'un bâtiment existant, de réaménager la disposition des espaces en minimisant les murs à déplacer tout en se rapprochant du cahier des charges avec le minimum de coût. Une interface homme/machine doit être développée comme une sur-couche d'ARCHiPLAN. On pourra par exemple désigner les murs ou parties de murs qu'on désirerait conserver (dont les murs porteurs).

7.3.8. ARCHiPLAN vu comme outil d'implantation industrielle

Bien qu'ARCHiPLAN ait été développé pour une utilisation en architecture, nous avons voulu voir s'il était adapté ou extensible à la problématique de l'implantation industrielle. Nous avons donc présenté ARCHiPLAN à un groupe d'ingénieurs[2] de Renault et effectué une visite au *Service des Méthodes Tôlerie* à boulogne-Billancourt.

L'évolution du patrimoine bâti, d'un constructeur automobile comme Renault, notamment de l'aménagement des usines de production, est pilotée par le schéma directeur de la société, c'est-à-dire sur des scénarios probables d'existence et de production de types de voitures donnés dans les 10 années à venir. Cet exercice hautement acrobatique est pourtant nécessaire pour faire évaluer peu à peu les usines

2. Nous avions rencontr par deux fois des ingénieurs de chez Renault de plusieurs services : implantation industrielle, logistique et informatique.

existantes vers des usines « idéales » ; l'idéal étant celui de la configuration des chaînes de production, de la performance des machines, de la taille des zones de stockage temporaires, de manière à accroître la rentabilité (plutôt que la productivité) ainsi que la flexibilité par rapport :

- aux demande évolutives de production,
- aux évolutions futures des produits (automobiles).

Cette évolution conjointe des produits et des moyens de production se fait progressivement par des réaménagements partiels des surfaces de production (typiquement tous les ans) et des investissements raisonnés dans certaines machines. Des ingénieurs en implantation passent du temps à produire des séries d'une dizaine de plans de surface évolutifs pour les différents scénario envisagés. On voit donc que l'aménagement d'une surface est conçue de manière évolutive sur 10 ans pour être à tout moment adaptée aux contraintes de production. Ce type de contraintes est sûrement extrêmement complexe à modéliser, d'autant plus qu'elle nécessiterait de connaître de très nombreuses données économiques.

D'autre contraintes ne seraient pas aisées à modéliser lors de chaque réaménagement d'une zone de l'usine. Il faut prendre en compte le déménagement et le stockage temporaire des machines de la zone au sein même de l'usine. Enfin, une autre contrainte difficile à appréhender, est que le dimensionnement d'une zone de stockage entre deux îlots de production d'une chaîne de production, dépend de la conception amont de la chaîne (performance des îlots, fiabilité de ces îlots et dimensionnement des zones de stockage précédentes).

En conclusion, nous avons constaté que les problèmes de l'implantation industrielle, c'est-à-dire la conception et l'évolution des usines de production, faisait intervenir des contraintes et des critères d'optimisation d'un tout autre niveau que ceux que nous avions considérés en architecture. Nous sommes tout à fait persuadés que cela pourrait faire l'objet d'une thèse intéressante.

7.3.9. Sur le plan technique

Il serait plus intéressant d'utiliser des langages compilés plus puissants comme C++. Le rapport de temps d'exécution varie de 1 à 30 avec les outils de développement que nous avons utilisés. La partie cohérence par arcs ne pose pas un gros problème de développement (deux à trois mois).

7.4. Conclusion

Nous avons montré à l'aide de nombreux tests la compétitivité d'ARCHiPLAN comparativement aux approches antérieures.

Nous avons montré qu'ARCHiPLAN pouvait traiter des problèmes de conception de taille moyenne en architecture.

Nous venons de présenter de très nombreuses améliorations possibles d'ARCHiPLAN, ainsi que des pistes pour leur résolution.

Nous espérons pouvoir apporter ces améliorations à ARCHiPLAN et pouvoir imaginer son industrialisation au dessus d'une plate-forme de CAO architecture.

Annexes

A.1. Rappel sur les problèmes de satisfaction de contraintes (CSP)[1]

Dans cette annexe, nous allons faire un rappel des principales notions de techniques de propagation de contraintes, à partir de résultats d'études bibliographiques de [Davis, 1985 ; Hanser, 1990 ; Jegou, 1991 ; Montanari, 1974 et 1989].

A.1.1. Définition

Un problème de satisfaction de contraintes (*CSP : Constraints Satisfaction Problems*) est défini par la donnée d'un ensemble de variables, chacune associée à un domaine fini de valeurs, et d'un ensemble de contraintes qui mettent en relation les variables et définissent l'ensemble des combinaisons de valeurs qui satisfont les contraintes. Une solution est une instanciation des variables qui satisfait toutes les contraintes. *Montanari* [Montanari, 1975] a été le premier à définir un *CSP* comme : $z = (X,D,C,R)$:

- (1) Un ensemble de n variables $X = \{x_1 .... x_n\}$
- (2) Un ensemble de n domaines finis $D = \{d_1 .... d_n\}$ où d_i est le domaine associé à la variable x_i.
- (3) Un ensemble de contraintes $C = \{ c_1 .... c_m\}$ où une contrainte c_i est définie par un ensemble de variables $\{x_{i1} .... x_{ini}\}$.
- (4) Un ensemble de relations $R = \{r_1 .... r_m\}$ où r_i est l'ensemble des combinaisons des valeurs qui satisfont la contrainte c_i.

• Exemple d'un *CSP*

Nous donnons ici la définition en terme de *CSP* d'un exemple de trois variables x, y, z. On peut définir ce problème (X,D,C,R) avec :

- Les variables $X = \{x, y, z\}$
- Les domaines $D = \{D(x), D(y), D(z)\}$ tels que :
  - $D(x)=[0,5]$,
  - $D(y)=[0,5]$,
  - $D(z)=[0,5]$.
- Les contraintes $C = \{c1 ,c2 ,c3\}$ telles que:
  - $c1 : x\ y = 3$
  - $c2 : x\ z = 3$
  - $c3 : y\ z = 3$

1. *Constraint Satisfaction Problem*

A.1.2. Recherche de solutions

A.1.2.1. Algorithme

Pour résoudre un problème de satisfaction de contraintes (énumération des solutions, optimisation), il existe différentes approches et méthodes. On peut considérer comme algorithme de base les procédures par essais et erreurs du type « *Backtracking* » (ou *retour arrière*), qui consistent à instancier les variables du problème. Lors de l'instanciation des variables, des tests de cohérence avec les contraintes sont réalisés entre cette instanciation et celles réalisées sur les variables antérieures, afin de vérifier si l'instanciation courante est cohérente. Si la variable que l'on vient d'instancier est la dernière variable dans l'ordre, l'instanciation courante constitue une solution au problème. Par contre, si une contrainte n'est pas satisfaite par la nouvelle instanciation, alors on choisit une autre valeur pour instancier la variable courante ; s'il n'en existe plus (elles ont été ainsi toutes essayées), un retour en arrière (ou *backtracking*) est réalisé sur l'instanciation de la variable qui précède la variable courante dans l'ordre afin d'essayer une autre valeur. Le *Backtracking* peut être employé pour chercher une solution ou toutes les solutions d'un problème.

A.1.2.2. Propagation de contraintes

La propagation de contraintes est un ensemble de techniques de déduction qui transforment un graphe de contraintes (cf. Figure A.1). Son objectif essentiel consiste à définir un problème équivalent au problème tel qu'il est posé, mais dont la nouvelle formulation devrait faciliter sa résolution. Les mécanismes sur lesquels repose la propagation de contraintes sont liés au notions de cohérence associées aux *CSP*.

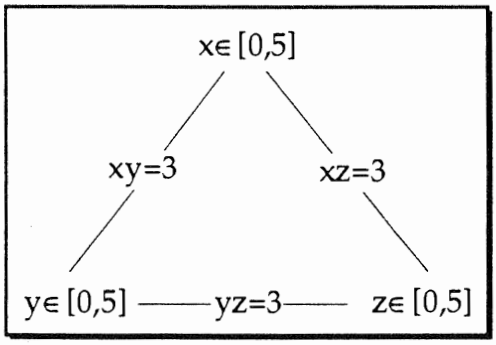


Figure A.1 : Graphe de contraintes

A.1.2.3. Notion de cohérence dans les *CSP*

La propagation que nous utiliserons correspond à de l'inférence de labels [Davis, 1985]. Elle conduit à supprimer des valeurs dans les domaines, valeurs dont on est assuré qu'elles ne figurent dans aucune des solutions. Ce procédé est parfois appelé filtrage du graphe. Ce filtrage simplifie le problème étant donné que la taille de l'espace de recherche se trouve réduite. ce filtrage s'opère jusqu'à un certain point pour conférer au graphe la propriété de *cohérence par arcs* ou, plus généralement, de *cohérence par chemin de rang n*.

La cohérence par arcs est un filtrage partiel qui assure une cohérence locale au niveau de tout couple de variables contraintes. Le filtrage par cohérence par arcs donne ici le graphe de la Figure A.2. En effet, la valeur $x_i=0$ par exemple est éliminée du domaine de x car il n'existe aucune valeur de y dans [0,5] telle que $x_i\, y_i=3$.

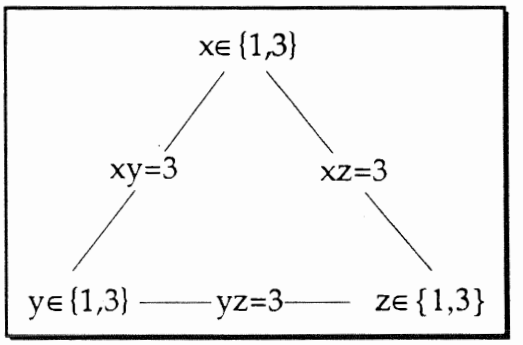


Figure A.2 : Graphe de contraintes aprés filtrage par *cohérence par arcs*

La définition est la suivante :

« *La variable contrainte x:[vx1,...,vxi,...] est cohérente par arc, si pour toute variable y:[vy1,...,vyj,...], et pour toute valeur de vxi de x, il existe au moins une valeur vyj du domaine de définition de y telle que l'ensemble des contraintes est respecté pour (vxi, vyj)* »

La *cohérence par chemin de rang n*, beaucoup plus coûteuse en temps (quand n>2), assure une cohérence moins locale, au niveau de tout n-uplet de variables contraintes. La définition de la *cohérence par chemin de rang n* est la suivante :

« *La variable contrainte x:[vx1,...,vxi,...] est cohérente par chemin de rang n, si pour tout (n-1)-uplet de variables y1,...,y(n-1), et pour toute valeur de vxi de x, il existe au moins un (n-1)-uplet de valeurs tel que l'ensemble des contraintes est respecté* »

La cohérence par chemin de rang 3 montrerait que $D(x)=D(y)=D(z)=\emptyset$ car les variables prises deux à deux laissent espérer des solutions (cf. Tableau A.1) mais aucun triplet

solution n'existe. En effet, lorsque *x* vaut *1*, y vaut *3* à cause de *c1*, *z* vaut *1* à cause de *c3* et *c2* devient incohérent.

	x	y	z
x y=3	1	3	
	3	1	
x z =3	1		3
	3		1
y z =3		1	3
		3	1

Tableau A.1 : Application de la cohérence par arcs au niveau des trois couples de variables

Il existe donc un compromis entre le temps passé à effectuer le filtrage du graphe et le temps passé à l'énumération des solutions car, plus le graphe est filtré, plus l'arbre de recherche est élagué. Le compromis se situe généralement au niveau de la *cohérence par arcs*.

A.1.2.4. Heuristiques utilisées

Les heuristiques en programmation par contraintes, contrairement à l'acception classique en IA et notamment dans les systèmes experts, ne réduisent en aucun cas le nombre de solutions obtenues. Leur rôle se limite juste dans le choix de l'ordre d'instanciation des variables du problème, et dans le choix de l'ordre d'instanciation des valeurs.

A.2. Glossaire

- *pièce.W* référence l'attribut contraint *W* de l'objet *pièce*,
- *D(pièce.W)* est le domaine de définition de *pièce.W*,
- *V(D(pièce.W))*=2 est l'instanciation de *pièce.W* à 2.
  Contrairement à la contrainte *pièce.W*=2, l'instanciation se « défait » lors d'un retour arrière,
- *V(D(pièce.W))*≡2 est, dans une contrainte démon, la condition de détection d'instanciation à 2 de *pièce.W*,
- → indique l'activation d'une contrainte "démon",
- % commentaires.

A.3. Contraintes de classe

A.3.1. Contrainte d'élimination des redondances d'orientation de la classe pièce

```
Contrainte          Elimine -Redondances     (IN : pièce)

lorsque  D(Pièce.L)≡(D(Pièce.W)
         → V(D(pièce.orientation))=0°
         %Les solutions correspondant à pièce.orientation = 90° seront toutes redondantes
lorsque          (ou    (Max(D(pièce.L)) > Max(D(pièce.W))
                        Min(D(pièce.L)) < Max(D(pièce.W))
                        Min(D(pièce.L)) > Min(D(pièce.W)))

                        (Max(D(pièce.L)) > Max(D(pièce.W))
                        Min(D(pièce.L)) < Max(D(pièce.W))
                        Min(D(pièce.L)) > Min(D(pièce.W))))
         Lorsque        V(D(Pièce.orientation))≡90°
                        → (ou  (V(pièce.W) ∈ [Min(D(pièce.L)) ... Min(D(pièce.L))[
                               V(pièce.L) ∈ [Min(D(pièce.W)) ... Min(D(pièce.L))[)

                               (V(pièce.W) ∈ ]Max(D(pièce.W)) ... Max(D(pièce.L))[
                               V(pièce.L) ∈ [Min(D(pièce.L))...Max(D(pièce.W))[))
lorsque          (ou    (Max(D(pièce.L)) > Max(D(pièce.W))
                        Min(D(pièce.L)) < Min(D(pièce.W)))

                        (Max(D(pièce.L)) > Max(D(pièce.W))
                        Min(D(pièce.L)) < Min(D(pièce.W)))))
         Lorsque        V(D(Pièce.orientation))≡90°
                        → (ou  (V(pièce.W) ∈ [Min(D(pièce.L)) ...Max(D(pièce1.W)))[
                               V(pièce.L) ∈ [Min(D(pièce.W)) ...Max(D(pièce1.W)))[))

                               (V(pièce.W) ∈ ]Max(D(pièce.W)) ...Max(D(pièce.L)))[
                               V(pièce.L) ∈ [Min(D(pièce.W)) ...Max(D(pièce.W)))[)))
fin Contrainte
```

A.3.2. Contrainte de classe de maintien de cohérence géométrique de la classe escalier à une volée

Contrainte **Maintien-de-cohérence-géométrique** (IN : escalier)

escalier.L-marche = escalier.L

lorsque V(D(escalier.orientation))≡0°
→ escalier.y-marche-de-départ = escalier.y2
escalier.y-marche-arrivée = escalier.y1
escalier.x-marche-de-départ = (escalier.x1 + escalier.x2)/2
escalier.x-marche-d'arrivée = (escalier.x1 + escalier.x2)/2,

lorsque V(D(escalier.orientation))≡90°
→ escalier.y-marche-de-départ = (escalier.y1 + escalier.y2)/2
escalier.y-marche-d'arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marche-de-départ = escalier.x1
escalier.x-marche-d'arrivée = escalier.x2,

lorsque V(D(escalier.orientation))≡180°
→ escalier.y-marche-de-départ = escalier.y1
escalier.y-marche-d'arrivée = escalier.y2
escalier.x-marche-de-départ = (escalier.x1 + escalier.x2)/2
escalier.x-marche-d'arrivée = (escalier.x1 + escalier.x2)/2,

lorsque V(D(escalier.orientation))≡270°
→ escalier.y-marche-de-départ = (escalier.y1 + escalier.y2)/2
escalier.y-marche-d'arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marche-de-départ = escalier.x2
escalier.x-marche-d'arrivée = escalier.x1,

fin Contrainte

A.3.3. Contrainte de classe de maintien de cohérence géométrique d'un escalier à deux volées

Contrainte Maintien-de-cohérence-d'un-escalier-à-deux-volées (IN: escalier)

escalier.L-marche= escalier.L

lorsque V(D(escalier.Position-marche-départ))≡gauche
lorsque V(D(escalier.orientation))≡0°
→ (escalier.y-marche-de-départ = escalier.y2
escalier.y-marche-arrivée = escalier.y1
escalier.x-marche-de-depart = (escalier.xl + escalier.x2)/2
escalier.x-marche-arrivée = (escalier.x1 + escalier.x2)/2)

lorsque V(D(escalier.orientation))≡ 90°
→ (escalier.y-marche-de-depart = (escalier.y1 + escalier.y2)/2
escalier.y-marche-arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marche-de-départ = escalier.xl
escalier.x-marche-arrive= escalier.x2)

lorsque V(D(escalier.orientation))≡180°
→ (escalier.y-marche-de-départ = escalier.y1
escalier.y-marche-arrivée = escalier.y2
escalier.x-marche-de-depart = (escalier.x1 + escalier.x2)/2
escalier.x-marche-arrive = (escalier.x1 + escalier.x2)/2)

lorsque V(D(escalier.orientation)) ≡270°
→ (escalier.y-marche-de-départ = (escalier.y1 + escalier.y2)/2
escalier.y-marche-arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marche-de-départ = escalier.x2
escalier.x-marche-arrivée = escalier.x1)

lorsque V (D(escalier.Position-marche-depart)) ≡droite
lorsque V(D(escalier.orientation -de-esc1))≡o°
→ (escalier.y-marche-de-depart = escalier.y2
escalier.y-marche-arrive = escalier.yl
escalier.x-marche-de-depart = (escalier.x1 + escalier.x2)/2
escalier.x-marche-arrive = (escalier.x1 + escalier.x2)/2)

lorsque V(D(escalier.orientation))≡ 90°
→ (escalier.y-marche-de-départ = (escalier.y1 + escalier.y2)/2
escalier.y-marche-arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marche-de-départ = escalier.xl
escalier.x-marche-arrivée= escalier.x2)

lorsque V(D(escalier.orientation)) ≡ 180°
→ (escalier.y-marche-de-départ = escalier.y1
escalier.y-marche-arrivée = escalier.y2
escalier.x-marche-de-départ = (escalier.x1 + escalier.x2)/2
escalier.x-marche-arrivée = (escalier.x1 + escalier.x2)/2)

lorsque V(D(escalier.orientation)) ≡ 270°
→ (escalier.y-marche-de-départ = (escalier.y1 + escalier.y2)/2
escalier.y-marche-arrivée = (escalier.y1 + escalier.y2)/2
escalier.x-marclle-de-dépard= escalierx2
escalierx-marche-arrivée = escalier.xl)

fin Contrainte

A.4. Contraintes spatiales

A.4.1. Les contraintes d'adjacence

A.4.1.1. La contrainte "*Adjacent-a-est* "

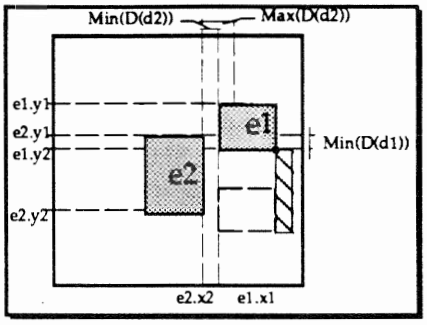


Figure A.3 : Le rectangle hachuré indique les positions permises du point (*x2*, *y2*) de l'espace *e1* par la contrainte *Adjacent-a-est*

Contrainte **Adjacent-a-est** (IN : e1, e2, d1, d2)

→ e1.x1=e2.x2+d2
e1.y2 ≥ e2.y1+ d1
e1.y1 ≤ e2.y2- d1)

fin Contrainte

Figure A.4 : La contrainte *Adjacent-a-est*

A.4.1.2. La contrainte "*Adjacent-au-sud*"

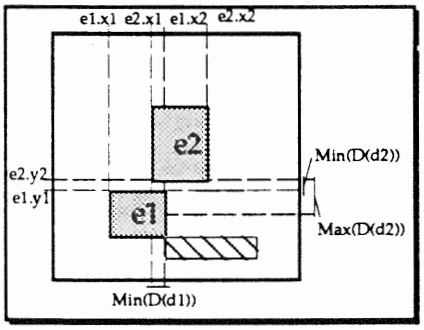


Figure A.5 : Le rectangle hachuré indique les positions permises du point (x2, y2) de l'espace e1 par la contrainte Adjacent-au-sud

Contrainte **Adjacent-au-sud** (IN : e1, e2, d1, d2)

→ e1.y1=e2.y2+d2
e1.x2≥e2.x1+d1
e1.x1≤ e2.x2-d1

fin Contrainte

Figure A.6 : La contrainte *Adjacent-au-sud*

A.4.1.3. La contrainte "*Adjacent-a-ouest*"

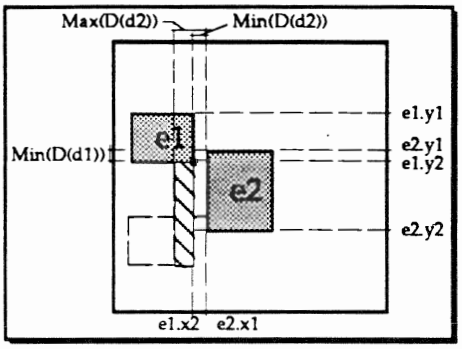


Figure A.7 : Le rectangle hachuré indique les positions permises du point (*x2*, *y2*) de l'espace *e1* par la contrainte *Adjacent-au-ouest*

Contrainte **Adjacent-a-ouest** (IN : e1, e2, d1, d2)

→ e1.x2=e2.x1-d2
e1.y2≥ e2.y1 +d1
e1.y1≤e2.y2-d1)

fin Contrainte

Figure A.8 : La contrainte *Adjacent-a-ouest*

A.4.1.4. La contrainte "*Adjacent-avec-marche-d'arrivée*"

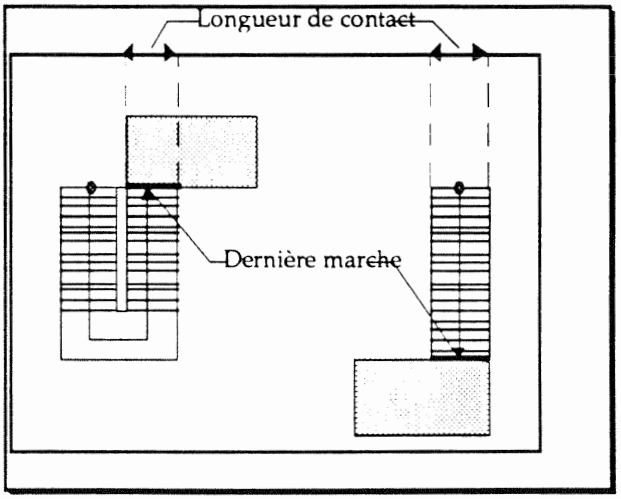


Figure A.9 : adjacence de la marche d'arrivée d'un escalier avec un couloir

Contrainte	**Adjacent-avec-marche-d'arrivée** (IN : escalier, pièce)
lorsque	escalier.orientation≡0°
→	escalier.y.marche.d'arrivée = pièce.y2
	x2.pièce ≥ escalier.x-marche-d'arrivée + (escalier.L-marche / 2)
	x1.pièce ≤ escalier.x-marche-d'arrivée - (escalier.L-marche)
lorsque	escalier.orientation≡90°

```
→          escalier.x.marche.d'arrivée = pièce.x1
           y2.pièce≥escalier.y-marche-d'arrivée + (escalier.L-marche / 2)
           y1.pièce≤escalier.y-marche-d'arrivée - (escalier.L-marche)
lorsque    escalier.orientation≡180°
→          escalier.y.marche.d'arrivée = pièce.y1
           x2.pièce≥escalier.x-marche-d'arrivée + (escalier.L-marche / 2)
           x1.pièce≤escalier.x-marche-d'arrivée - (escalier.L-marche)
lorsque    escalier.orientation≡270°
→          escalier.x.marche.d'arrivée = pièce.x2
           y2.pièce≥escalier.y-marche-d'arrivée + (escalier.L-marche / 2)
           y1.pièce≤escalier.y-marche-d'arrivée - (escalier.L-marche)
fin Contrainte
```

Figure A.10 : Contrainte d'adjacence avec l'escalier pour la descente

A.4.2. Les contraintes d'adjacence avec l'espace de placement

A.4.2.1. La contrainte "*Sur-la-façade-est*"

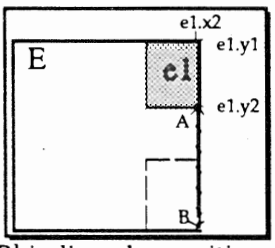


Figure A.11 : Le segment [AB] indique les positions permises du point (*x2*, *y2*) de l'espace *e1* par la contrainte *Sur-la-façade-est*

```
Contrainte  Sur-la-façade-est (IN : e1, E)
→ e1.x2=E.x2
  e1.y1≥E.y1
  e1.y2≤E.y2
fin Contrainte
```

Figure A.12 : la contrainte *Sur-la-façade-est*

A.4.2.2. La contrainte "*Sur-la-façade-sud*"

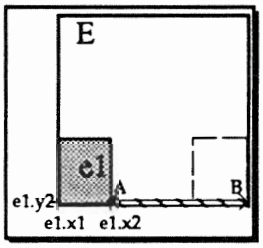


Figure A.13 : Le segment [AB] indique les positions permises du point (*x2*, *y2*) de l'espace *e1* par la contrainte *Sur-la-façade-sud*

Contrainte Sur-la-façade-sud (IN : e1, E)

→ e1.y2=E.y2
e1.x1≥E.x1
e1.x2≤E.x2
fin Contrainte

Figure A.14 : La contrainte *Sur-la-façade-sud*

A.4.2.3. La contrainte "*Sur-la-façade-ouest*"

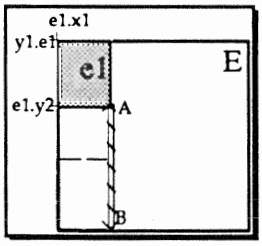


Figure A.15 : Le segment [AB] indique les positions permises du point (*x2*, *y2*) de l'espace *e1* par la contrainte *Sur-la-façade-ouest*

Contrainte Sur-la-façade-ouest (IN : e1, E)
→ e1.x1=E.x1
e1.y1≥E.y1
e1.y2≤E.y2
fin Contrainte

Figure A. 16 : La contrainte *Sur-la-façade-ouest*

A.4.3. Les contraintes d'orientation relative

A.4.3.1. La Contrainte "*A-est-de* "

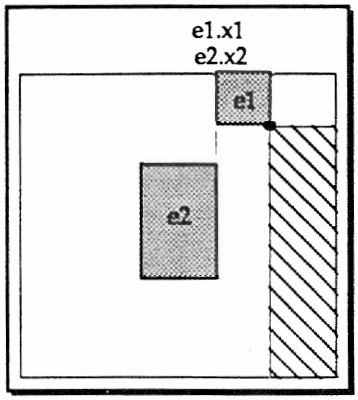


Figure A.17 : Positions permises pour le point (*x2,y2*) de l'espace *e1* par la contrainte *A-est-de*

Contrainte A-est-de (IN : e1, e2)
→ e1.x1≥e2.x2
fin Contrainte

Figure A.18 : La contrainte *A-est-de*

A.4.3.2. La Contrainte "*Au-sud-de*"

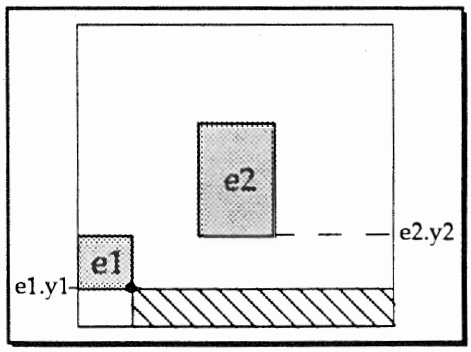


Figure A.19: Positions permises pour le point (*x2,y2*) de l'espace *e1* par la contrainte *Au-sud-de*

Contrainte Au-sud-de (IN : e1, e2)
→ e1.y1≥e2.y2
fin Contrainte

Figure A.20 : La contrainte *Au-sud-de*

A.4.3.3. La Contrainte "*A-ouest-de*

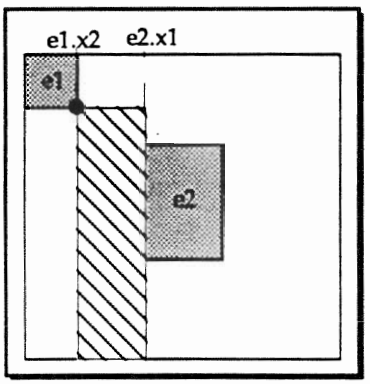


Figure A.21: Positions permises pour le point ($x2$,$y2$) de l'espace *e1* par la contrainte *A-Ouest-de*

Contrainte A-Ouest-de (IN : e1, e2)
→ e2.x1≥e1.x2
fin Contrainte

Figure A. 22 : La contrainte *A-Ouest-de*

A.5. Utilisation d'ARCHiPLAN en ordonnancement

Nous avons adapté ARCHiPLAN à un problème d'ordonnancement de tâches proposé et mis en œuvre sur ARCHiPLAN par *Safia Keddad* [Keddad, 1994]. Il s'agit d'ordonner 10 tâches (t1 à t10) sur deux machines parallèles (M1 et M2). Les tâches comme les machines sont représentées par des rectangles de hauteur identique et dont la longueur correspond à la durée, le cahier des charges est spécifié ci-dessous :

- t1, t2, t4, t5, t7 et t9 sont affectées à la machine M1, (équivaut à la contrainte *Dans*),
- t3, t6, t8 et t10 sont affectées à la machine M2,
- Les tâches se déroulent dans des *fenêtres de temps* (la fenêtre de temps impose que la tâche débute à partir du temps td et soit finie avant tf. La syntaxe est :

  fenêtre_temps(tâche, td, tf)

  Cette contrainte correspond dans ARCHiPLAN à imposer un minimum à la variable *x1* d'une *pièce* et un maximum à la variable *x2* de la même *pièce*. les contraintes sont donc :

  - fenêtre_temps (t1, 0 , 15)
  - fenêtre_temps (t2, 13, 25)
  - fenêtre_temps (t3 , 25 ,70)
  - fenêtre_temps (t4 , 10 ,15)
  - fenêtre_temps (t5 , 23 ,50)

- fenêtre_temps (t6 , 0, 20)
- fenêtre_temps (*t7*, 50, 80)
- fenêtre_temps (*t8*, 10, 30)
- fenêtre_temps (*t9* , 30, 60)
- fenêtre_temps (*t10*, 60, 100)

- Des *contraintes de lancement* sont imposées. Il s'agit de respecter un délai minimum entre la fin de la tâche *ti* et le début *tj*. Le délai Tp_{ij} est imposé lorsque *ti* est avant *tj* et le délai Tp_{ji} est imposé lorsque *tj* est avant *ti*. La syntaxe est donc

lancement (ti, tj, Tp_{ij} , Tp_{ji})

La contrainte de lancement équivaut à imposer une distance minimale entre deux espaces e1 et e2 qui est différente dans le cas où e1 est à l'Est ou à l'Ouest de e2.

- lancement (*t1*, *t4*, 1, 10)
- lancement (*t4*, *t2*, 1, 20)
- lancement (*t2*, *t5*, 1, 10)
- lancement (*t5*, *t9*, 1, 20)
- lancement (*t9*, *t7*, 1, 30)
- lancement (*t6*, *t8*, 1, 30)
- lancement (*t8*, *t3*, 1, 20)
- lancement (*t3*, *t10*, 1, 30)

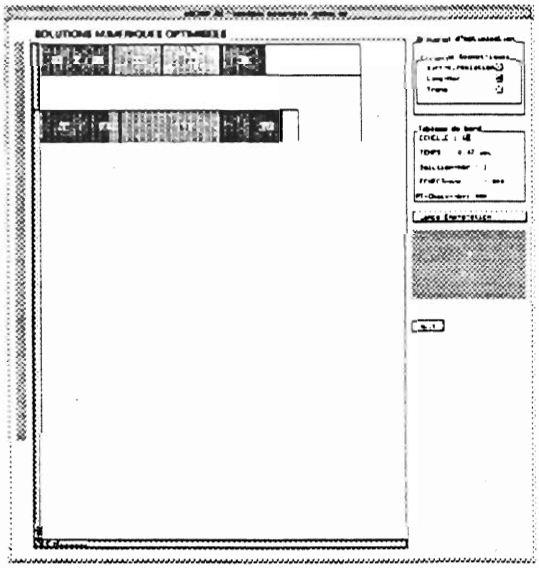


Figure A23 : résultats obtenus sur ARCHiPLAN pour le problème d'ordonnancemment

Nous avons avons illustré dans la Figure A.23 la première solution minimisant les fenêtres de temps. Cette solution a été obtenue en 0,47 seconde (temps CPU).
Cette première tentative d'adaptation d'ARCHiPLAN à un problème d'ordonnancement de tâches ayant été fructueux, nous envisageons d'approfonfir ce domaine.

Bibliographie

1. **Aérospatiale (Cannes)**. Contraintes géométriques sur l'aménagement des panneaux de satellites. *Journée contraintes géométriques*. INRIA Sophia-Antipolis, Décembre 1992.

2. **A. Aggoun, N. Beldiceanu.** Extending CHIP in Order to Solve Complex Scheduling and placement Problems. *In Journée française de la programmation logique*, Marseille, 1992.

3. **C. Alexander**. De la synthèse de la forme. Dunod, Paris 1971.

4. **J.M. André.** Vers un système d'aide intelligent pour l'aménagement spatial : CADOO. *In actes du Deuxième colloque International d'Intelligence Artificiel de Marseille (CIAM 86)*, pages: 31-48, 1986.

5. **M.E. Arnaud.** *La composition en Architecture*. Édition de l'École Centrale des Arts et Manufactures, Paris, 1928.

6. **C.A. Baykan, M.S. Fox.** Constraint Satisfaction Techniques for Spatial Planning. In Intelligent CAD Systems III : Practical experience and Evaluation. Page: 187-204, 1991.

7. **H. Boccond-Gibod.** *Link Concepts for PDMS: Preliminary Study*. Rapport de recherche DER/EDF, Clamart, 1993.

8. **H. Boccon-Gibod, L. Lehmann et G. Thibault.** Modélisation de la cour carrée du Louvre. Rapport techniques, (groupe CAO) EDF/DER, 1993.

9. **Ph. Boudon**. Sur l'espace architectural. Edition Dunod, Paris, 1971.

10. **O. Bourdin et Y. Godillot**. Comparaison entre la conception variationnelle et paramétrique. In actes de Micad'91, édition Hermès, Paris, 1991.

11. **Centre George Pompidou.** *Le Corbusier*. Édition du Centre George Pompidou, Paris, 1987.

12. **P. Charman.** *Gestion des contraintes géométriques pour l'aide à l'aménagement spatial*. Thèse de doctorat de l'École Nationale des ponts et chaussées, novembre 1995.

13. **J.C. Chung et M.D. Schussel.** Comparison of Variational and Parametric Design. *Revue International de CFAO et d'Infographie*. (4): 81-101, 1989.

14. **P. Coad et E.Yourdon.** *Object-Oriented Design.* Yourdon Press, New Jersey, 1991.

15. **P. Coad et E. Yourdon.** *Analyse Orientée Objet*. Édition Dunod, Paris, 1992.

16. **A. Colmerauer**. Une introduction à PROLOG III. *In actes des 10èmes journées internationales : les systèmes experts & leurs applications*, Avignon, 1990.

17. **R.D. Coyne et J. Gero**. *Knowledge-Based Design Systems*. Addison-Wesley Publishing Company, 1991.

18. **CSTB**. Échange de données techniques dans la construction. *In actes du Colloque*. CSTB, Paris, 1992.

19. **E. Davis.** Constraint propagation With Interval Labels. *Artificial intelligence.* (32): 281-331, 1987.

20. **DICOBAT**. Dictionnaire général du bâtiment. Édition Arcature, Paris 1991.

21. **S. Donikian.** Towards a Declarative Method for 3D Scene Sketch Modeling. *Journée contraintes géométriques*, INRIA, décembre 1992.

22. **J. Dufau et A. Messabhia.** Définition de la technologie de bâtiments dans un contexte de CAO : Habillage par commandes et système expert de déduction de solution techniques. EUROP'IA, 1988.

23. **C. et M. Duplay**. Méthode illustrée de création architecturale. Édition du Moniteur, Paris, 1985.

24. **J.N.L Durand.** *Précis des Leçons d'architecture.* Imprimerie du Roi, Paris, 1801.

25. **F. DuVerdier**. Solving Geometric Constraint Satisfaction Problems for Spatial Planning. *In Actes of 13th International Joint Conference on Artificial Intelligence.* Chambéry (France), 1993.

26. **C. Eastman**. Automation Space Planning. *Artificial Intelligence*,(4): 41-64, 1973.

27. **D. Elalouf.** Christofer Alexander. *Architecture d'Aujourd'hui*. (174) : 54-63, 1974.

28. **U. Flemming**. A generative Expert System for the Design of Building Layouts. *Final Report, Engineering Design Reasearch Center*, Carnegie Mellon University, EDRC 48-15-89, 1988.

29. **U. Flemming, C.A. Baykan, R.F. Coyne et M.S. Fox**. Hierarchical Generate and Test vs constraint-directed search. AI in Design'92, J.S. Gero (ed) Kluwer 1992, 817-838.

30. **J. Fletcher.** A program to solve the pentomino Problem by the Recursive Use of Macros. Communications of the ACM, 8(10): 621-623, octobre 1965.

31. **P. Galli et U. Cugini.** System for Parametric Definition of Engineering Drawings. *In actes de Micad. 85*, Paris, 1985.

32. **G. Goldschmidt.** Criteria for Design Evaluation: A Process-Oriented ParadIgm. *In Evaluating and Predicting design Performance.* Y.E. Kalay Editor, a Wiley-Interscience Publication, New York. 67-79, 1992.

33. **J.F. Goulette.** Astragale, une maquette d'étude sur l'articulation entre la phase d'esquisse et l'instrumentation du projet d'architecture. *In Actes de Europ'IA.* National Technical University of Athens, 1991.

34. **GSD.** *Synthèse des modèles conceptuels développés dans le cadre de la recherche bâtiment en France.* Plan Construction et architecture, 1991.

35. **F. Guéna**. Un raisonnement par analogies pour la résolution des problèmes de conception qui requièrent de l'innovation. *In Actes de 01 Design*, Marrakech, Maroc, 1992.

36. **F. Guéna.** Representation et utilisation de pré-solutions pour la resolution des problèmes de conception. *In Actes de Representation par Objets : Le point sur la recherche et les applications.* La Grande Motte, le 22-23 juin 1992.

37. **H. Chaouch et E. Cocquebert**. State of the Art and Evolution in Feature-Based Modeling. *Revue International de CFAO et d'Infographie.* (2): 169-200, 1992.

38. **S. Hanrot.** *Modélisation de la connaissance architecturale pour un outil de CAO intelligent.* Édition la Recherche. Paris, 1989.

39. **J.M. Hanser**. CFAO et propagation de contraintes, technologie, enjeux et étapes industrielles. *In Actes de Micad*, édition Hermès, Paris, 1990.

40. **R.M. Haralick** et **G.L. Elliott.** Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, (14): 263-313, 1980.

41. **C. Honda.** A Floor Planning System Using Constraint Logic Programming. *In the Second International Conference on the Practical Application of Prolog*, London, 1994.

42. **K. Hua et B. Faltings.** Exploring Case-Based Building Design-CADRE. *In Artificial intelligence For Engineering Design, Analysis and Manufacturing.* (2): 135-144, 1993.

43. **Ilog.** PECOS Manuel de référence, Version 1.2, Paris, 1991.

44. **Ilog.** Lelisp Manuel de référence, Version 15.2, Paris, 1991.

45. **P. Jegou.** *Contribution à l'étude des problèmes de satisfaction de contraintes : Algorithmes de propagation et de résolution. Propagation de contraintes dans les réseaux dynamiques.* Thèse de l'Université de Montpellier II, France, 1991.

46. **J.H. Jo et J.S. Gero.** Space Layout Planning Using an Evolutionary Approach. *In actes CAAD Futurs'95*, Singapore, 1995.

47. **J.P. Jung.** Un nouveau modèle pour la représentation de contraintes complexes en CAO. *Revue Internationale de CFAO et d'Infographie.* (9): 9-23, 1994.

48. **S. Kedad.** Ordonnancement de production à machines parallèles. 2ème colloque des thésards de l'Intergroupe des Ecoles Centrales. Ecole Centrale Paris (France), 1994.

49. **L.B. Kovacs.** Knowledge Based Floor Plan Design by Space Partitionning: A Logic Programming Approach. *Artificial Intelligence in Engineering.* 6(4): 162-185, 1991.

50. **R.H. Kurland.** La modélisation pilotée par les variables. *Bureaux d'études : le magazine des concepteurs.* (85) : 34-37, 1993.

51. **J.L. Laurière.** Un langage et un programme pour résoudre et énoncer des problèmes combinatoires : ALICE. Thèse d'état de l'Université Paris 6, 1976.

52. **J.C. Lebahar.** *Étude en CFAO, Architecture et bâtiment.* Édition Hermès. Paris, 1985.

53. **J.C. Lebahar.** *Le dessin d'architecte.* Édition Roquevaire : Parenthèse, 1983.

54. **Le MONITEUR.** Informatique et architecture. n°18, février 1991.

55. **R. S. Ligett**. Designer-Automated Algorithm Partnership: An Interactive Graphic Approach to Facility Layout. *In Evaluating and Predicting design Performance*. Y.E. Kalay Editor, a Wiley-Interscience Publication, New York. 101-123, 1992.

56. **R. S. Ligett et W. J. Mitchell**. Optimal Space Planning in Practice. *In Computer Aided Design*, 13(5):277-288, 1981.

57. **R. S. Ligett, W. J. Mitchell et M. Tan**. Multilevel Analysis and Optimization of Design. *In Evaluating and Predicting Design Performance*. Y.E. Kalay Editor, a Wiley-Interscience Publication, New York. 251-269, 1992.

58. **A. Lurça.** Formes, Composition et Lois d'harmonie : éléments d'une science de l'esthétique architecturale. Édition Vincent Fréal & C^{ie}, Vol. 2 et 4., Paris, 1955.

59. **R. Maculet.** *Représentation des connaissances spatiales (Algèbre de Manhattan) et raisonnement spatial avec contraintes*. Thèse de doctorat de l'Université Paris 6, France, 1991.

60. **R. Maculet, B. Medjdoub et H. Boccon-Gibod.** *Civil Structure in PDMS*. Rapport de recherche ECP/EDF, 1992.

61. **C. Manago.** *Étude d'un système expert appliqué à la résolution de problèmes en architecture*. Thèse de doctorat de l'Université Paris 6, France, 1984.

62. **J.P. Maroy.** Description d'espaces pour les problèmes d'allocation spatiale. Institut de l'Environnement, Centre de mathématiques, Méthodologie et Informatique, 1973.

63. **B. Medjdoub**. Towards Aid in Preliminary Design in Architecture: ARCHiPLAN. *In Actes de Computing in Civil and Building Engineering: Proc.* A.BALKEMA publisher, Berlin, 1995.

64. **B. Medjdoub et R. Maculet.** Maintien de cohérence dans un système constructif du bâtiment. *In actes de 01 Design*, Gamarth (Tunisie), 1993.

65. **B. Medjdoub et B. Yannou**. ARCHiPLAN : un outil de conception fonctionnelle en implantation industrielle. *In actes de IDMME-96 : First International Conference on Integrated Design and Manufacturing in Mechnical Design*. Nantes, France, 1996.

66. **B. Medjdoub et B. Yannou**. A Functional Approach in Architectural CAD Softwares : ARCHiPLAN. In ICTCSE'96. *International Conference on Information Technology in Civil and Structural Engineering Design*, Glasgow, Ecosse, 1996.

67. **W. J. Mitchell, J. P. Steadman et R.S. Ligett**. Synthesis and Optimisation of Small Rectangular Floor Plans. *In Environment and Planning B*, 3:37-70, 1976.

68. **M. Momessin et G. Sauce**. Un sytème expert de déduction de conséquences multitechniques associé à une modification dans un contexte de CAO bâtiment. In actes de Europ'IA 88, 1988.

69. **U. Montanari**. Networks of Constraints: Fundamental Properties and Application to Picture Processing. *Information Science*.(7): 95-132, 1974.

70. **U. Montanari et Rossi**. Fundamental Properties of Networks of Constraints: a New Formulation. Kanal and Kumar V. editor, Springer-Verlag. 1989.

71. **A. Mukerjee et S.E. Bratton**. A Qualitative Model for Spatial Learning. *In actes de Avignon'91*. Avignon, 1991.

72. **E. Neufert**. *La coordination dimensionnelle dans la construction*. Édition Dunod, 1967.

73. **E. Neufert**. *Les éléments des projets de construction*. 6eme Édition, Dunod, 1983.

74. **A. Paoluzzi et C. Sansoni.** Programming Language for Solid Variational Geometry. *Computer Aided Design*, vol 24, 1992.

75. **Perouse de Montélis**. *Vocabulaire de l'architecture*. Imprimerie nationale, Paris, 1972.

76. **C.E. Pfefferkorn**. A heuristic Problem Solving Design System for Equipement or Furniture Layouts. *Communication of the ACM*, 18(5): 286-297, 1975.

77. **S. Pimont et M. Tollenaere.** Modèles et techniques pour l'aménagement spatial. *Revue Sciences et Techniques de la conception*, (2): 97-123, 1993.

78. **P. Puget**. On the Satisfiability of Symmetrical Constrained Satisfaction Problems. *In Methodologies for Intelligent Systems: Proc, of the 7th International Symposium ISMIS-93*. Springer-Verlag, 1993.

79. **P. Puget et P. Albert.** PECOS, programmation par contraintes orientée objets. *Génie logiciel & systemes experts*. 100-105, 1991.

80. **P. Quintrand, J. Autran, M. Florenzano, M. Fregier et J. Zoller.** *La conception assistée par ordinateur en Architecture.* Édition Hermès, Paris, 1985.

81. **P. Quintrand et J.C Paul.** Perspectives de la CAO en architecture. *Revue Internationale de CFAO et d'Infographie.* 7(1) : 63-69, 1992.

82. **RAUC**. *Sciences et techniques du bâtiment, 5000 mots clés expliqués.* Édition du moniteur, Paris, 1983.

83. **M. Renaud**. Bandit : un système expert en architecture. *In actes deEurop 'IA 88.* 1988.

84. **J.A. Roach**. The Rectangle Placement Language. *In IEEE Proc*, 1984.

85. **J. Roth et R. Hashimshony**. Algorithms in Graph Theory and their Use for Solving Problems in Architectural Design. *Computer Aid Design*. 20:373-381, 1988.

86. **D. Sacerdoti.** Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence*. (5):115-135, 1974.

87. **A. Saffo et S. Kharrufa**. Developing CAD Techniques for Preliminary Architectural Design. *Computer-Aided Design*, (20): 581-588, 1988.

88. **D. Sciamma.** Charme : des contraintes aux solutions. *Génie logiciel & systèmes experts*. 19 :100-102, 1990.

89. **H.A. Simon.** Search and Reasoning in Problem Solving. *Artificial Intelligence.* (21):7-29, 1983.

90. **I. Smith.** Creative Design Objects From Cases For Interactive Spatial Composition. To be presented at the 4th International Conference on Artificial Intelligence in Design (AID'96) Stanford University, 1996.

91. **Stefik**. Planning with Constraints (Molgen:Part1). *In Artificial Intelligence.* (16): 110-140, 1981.

92. **C. Tong**. Towards an Engineering Science of Knowledge-based Design. *In Artificial Intelligence in Engineering*, 2(3) :133-166, juillet 1987.

93. **B. Trousse.** Using AI in Complex System Design: Implications on Building ICAD Tools. *Revue Internationale de CFAO et d'Infographie.* 8(3) : 303-335, 1993.

94. **F. Verluise.** *Méthodologie de formulation des règles d'expertises pour les systemes constructifs: expérimentation du système expert snark sur le système construction S.E.S.* Plan Construction et Architecture, 1986.

95. **R. Villamayor.** *Définition Formelle d'un générateur de solutions d'allocation spatiale en deux dimensions.* Thèse de doctorat de l'Institut National Polytechnique de Grenoble, France, 1980.

96. **A. Wiezel et R. Becker.** Integration of Performance Evaluation in Computer-Aided Design. *In Evaluating and Predicting Design Performance.* Y.E. Kalay Editors, A Wiley-Interscience Publication, 171-181, New York.1992.

97. **C. Xuejun et H. Zhinjun.** Automated Design of House-Floor Layout with Distributed Planning. *Computer Aided Design.* (22):213-222, 1992.