Conception créative assistée par ordinateur : un modèle d'intelligence interactive

Denis Lalanne

Jury :

Président : Prof. Hannes Bleuler

Directeur de thèse : Dr Pearl Pu

Rapporteurs :

Prof. Reymond Clavel

Dr Catherine Garbay

Dr Markus Stolz

Dr Peter Struss

Prof. Paul Xirouchakis

January 31, 1999

Remerciements

Je tiens à remercier tout particulièrement mon directeur de thèse, le docteur Pearl Pu, pour son soutien intellectuel, pour sa confiance en mes idées et pour son application à m'enseigner la rigueur scientifique.

Je souhaite également remercier le professeur Reymond Clavel qui m'a guidé dans le monde de la conception et m'a conté ses propres expériences.

Je n'oublie pas le docteur Catherine Garbay qui m'a, la première, intéressé à la recherche ainsi que le docteur Markus Stolze de l'institut de recherche d'IBM Zurich qui m'a écouté attentivement à la fin de ma thèse et qui m'a donné des références intéressantes.

Je remercie également mon collègue George Mellisargos pour les nombreuses discussions stimulantes que nous avons eues ensemble.

Je souhaite aussi remercier Omar Tazi qui a rendu l'ambiance de travail du bureau agréable. Il a apporté de la fraîcheur dans ma recherche en recodant partiellement le système Comind avec le langage Java afin de favoriser la coopération entre plusieurs groupes de concepteurs (projet MicroCE). Je remercie aussi les étudiants de l'Ecole Polytechnique qui m'ont aidé dans mes travaux. Frank Schnyder a grandement participé à la réalisation de l'assistant ShowCase et de l'assistant Brainstorming.

Je tiens aussi à remercier le ≪Fonds national suisse de la recherche scientifique≫ pour son support financier ainsi que tous les membres de mon institut pour leurs différentes aides.

Je remercie enfin mon entourage pour l'amitié et l'amour qu'il me porte et pour le soutien tout au long de mon travail. Je pense à mon père, ma mère, ma soeur, mes frères et à leurs enfants ainsi qu'à Adriana, Alain, Elodie, Giulia, Ioanna, Lorenzo, Luc, Lue, Nelly, Nicolas, Pascal, Patrick, Rachel, Véronique et Zoran.

Résumé

L'objectif de cette thèse est d'établir un modèle d'interaction entre l'homme et la machine dans une tâche de raisonnement. Cette interaction favorisera la collaboration de l'intelligence humaine avec celle de la machine. Nous espérons ainsi créer une synergie aboutissant à une intelligence interactive. Notre hypothèse principale est que pour réaliser une telle collaboration, il est important de considérer la complémentarité de l'homme et de la machine.

Afin de valider cette hypothèse, nous avons choisi de traiter la conception créative. Ce type de résolution de problèmes nécessite aussi bien la participation de l'humain, pour ses facultés créatives, son intuition artistique et son riche répertoire de connaissances, que de celle de la machine, pour ses capacités de calcul, de visualisation et de mémoire. Nous proposons donc dans cette thèse un modèle informatique basé sur la complémentarité et la collaboration entre l'humain et la machine, modèle qui permettra d'améliorer les facultés des concepteurs.

Nous montrons qu'une architecture ouverte, distribuée entre plusieurs assistants de calcul et réflexifs, interagissant avec le concepteur au travers de visualisations, ne bride pas sa créativité et peut même la stimuler.

Nous exposons dans cette thèse notre point de vue à propos de la conception créative. Nous présentons ensuite notre système COMIND (pour COmputer aided creativity and Multicriteria optimization IN Design) et montrons comment il peut augmenter les facultés calculatoires, perceptives et attentionnelles d'un concepteur. Nous présentons ensuite en détail tous les assistants de notre système ainsi que les nouvelles métaphores de visualisation que nous avons inventées. Enfin, nous présentons deux scénarios d'utilisation de notre système et son évaluation.

Abstract

The goal of this thesis is to establish a model of interaction between human and machine in a reasoning process. This interaction, therefore, has to facilitate the collaboration in problem resolution by emphasizing and compensating the cognitive and perception powers of human and machine. We call such a synergy interactive intelligence. Our main hypothesis is that in order to realize such a collaboration, it is important to analyze and stress on the difference, rather than similarity, between human and machine skills.

In order to validate this hypothesis, we have chosen to study creative design, a problem solving process that requires not only the human's creative skills, artistic intuitions, and knowledge, but also the machine's calculation, visualization, and memory power. Our analysis of creative design has led us to propose a computational model of collaboration based on artificial intelligence techniques for generating design solutions and communicating these solutions to users via a set of new visualization tools.

This model has an open architecture, distributed around the user, and shared between computational and reflexive machine assistants. Thus our user-involved AI system employs visualization to confirm human's problem solving path and thus mirror the process of collaboration.

We present, in this thesis, our point of view about creative design and our belief in the human and machine complementary intelligence. We present in detail all the assistants and the new visualisation metaphors of our system. A special focus is made on two assistants, one supporting the discovery of the optimal solutions in an under-constrained problem and the other one helping to visualize conflicts and relax them in an over-constrained problem. Finally, we'll present two working examples using our system Comind and give an evaluation of some features of it.

Table des matières

Chapitre 1

Introduction

1.1 Motivations

En ne considérant pas les facteurs humains des systèmes intelligents, nous risquons de construire de gros logiciels capables de résoudre des problèmes complexes, mais incapables de communiquer avec les utilisateurs. Notre objectif principal est de concevoir des interfaces qui rendent utilisables par le plus grand nombre les puissants algorithmes de l'Intelligence Artificielle. Nous désirons créer un type d'≪intelligence interactive≫ (voir section 2.3, page 11), une synergie entre l'homme et la machine au niveau du raisonnement, question peu abordée jusqu'à présent. Afin de créer une telle synergie, il faut établir une collaboration efficace entre l'homme et la machine.

La collaboration est un sujet particulièrement intéressant dans le domaine de l'interaction homme-machine car elle nécessite l'analyse de la complémentarité de l'humain et de la machine, plutôt que la similarité. Historiquement, cette approche est nouvelle et elle nous semble une contribution importante à apporter dans une tâche de raisonnement. Nous avons choisi de soutenir plus spécifiquement la conception car c'est un processus de pensée difficile qui nécessite aussi bien la participation de l'humain, pour ses facultés créatives, son intuition artistique et son riche répertoire de connaissances, que de celle de la machine, pour ses capacités de calcul, de visualisation et de mémoire. Aussi, la conception a toujours été le terrain d'expérimentation des nouvelles théories.

La conception n'est pas la description de ce qui est, elle est l'exploration de ce qui pourrait être [Mitchell, 1990]. C'est un des domaines d'application les plus critiques qui nécessite l'utilisation des machines du fait des calculs impliqués. L'humain et la machine doivent donc pouvoir collaborer, s'enrichissant mutuellement de leurs facultés complémentaires. Cependant, la plupart des outils existants de CAO (Conception Assistée par Ordinateur) et de ICAO (CAO Intelligente) ne permettent pas à

des concepteurs d'exprimer leurs qualifications créatives ([Lalanne and Pu, 1996]). Ils imposent plutôt aux créateurs une structure algorithmique pré-définie.

Ces raisons motivent notre désir d'apporter, dans cette thèse, les éléments qui permettront de construire une véritable collaboration homme-machine dans la résolution de problèmes, et, plus spécifiquement, dans la conception de type configuration. Afin de construire ce modèle, nous répondrons aux questions suivantes:

- comment caractériser ce type de conception?
- quelles étapes de la conception créative sont soutenues?
- quelles sont les métaphores nécessaires au niveau de l'interface pour réaliser une collaboration homme-machine dans une tâche de raisonnement?

1.2 Conception configurable

Dans cette thèse, nous traitons un type particulier de conception. Les conceptions que nous abordons doivent être configurables. Une conception de type configuration consiste à rechercher l'espace des combinaisons possibles des constituants d'un produit (figure 1.1). Par exemple, ce genre de conception peut être: Un catalogue de solutions (description fonctionnelle), la conception d'un lotissement, la configuration d'une chaine HiFi, etc. Nous verrons dans le chapitre 4 comment nous formalisons ce genre de configuration sous la forme d'un CSP (Constraint Satisfaction Problem).

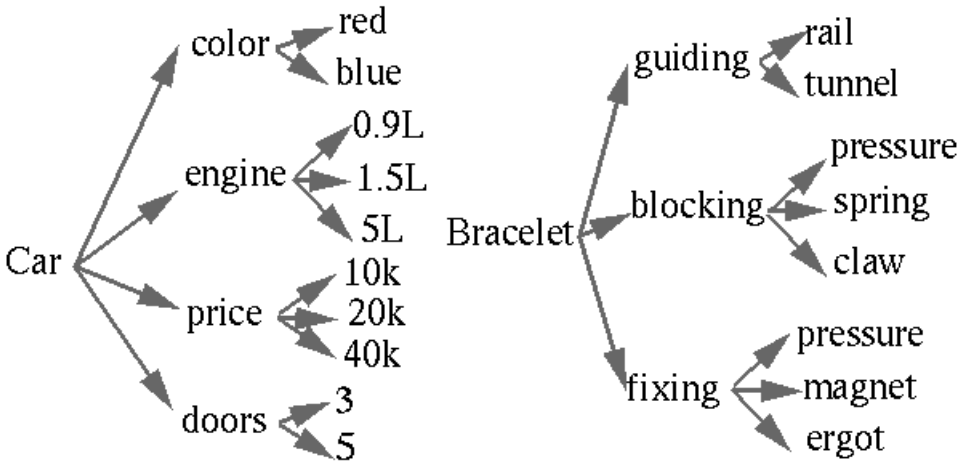

Figure 1.1: Une conception de type configuration est décomposable en constituants. L'espace de leurs combinaisons correspond aux concepts possibles.

1.3 Conception créative: une nouvelle taxinomie

Nous ne croyons pas en une taxinomie claire de la conception parce que ses frontières sont floues et qu'il est généralement difficile de classifier une conception spécifique. Dans la plupart des cas, un concepteur doit faire face à des processus cognitifs aussi bien routiniers[1], qu'innovateurs ou créatifs. Dire qu'un monde fermé correspond à un processus automatique de la pensée et qu'un monde ouvert correspond à une pensée innovatrice ou créative n'est pas suffisant. C'est la complexité de chacune des étapes qui permet de distinguer une conception d'une autre. Notre propre taxinomie vient de l'observation du processus cognitif créatif. Nous nous sommes intéressé à deux types de processus cognitifs de la conception créative: l'optimisation multicritère[2] et la résolution de conflits. Ces deux types de processus cognitifs sont eux-mêmes issus de deux types de problèmes extrêmement distincts: les problèmes sous-contraints et les problèmes sur-contraints. Nous montrons dans cette thèse comment soutenir ces deux processus. Dans les problèmes sous-contraints, trop de solutions sont possibles. Le concepteur doit jongler avec différents critères, les balancer, et faire des compromis pour trouver la solution optimale. Cependant, du fait du nombre souvent important de critères, les choix sont difficiles. Lorsque le problème est sur-contraint, il n'y a aucune solution. Le concepteur doit trouver les conflits dans la définition du problème et les résoudre. Cette tâche nécessite deux sous-processus cognitifs: premièrement la vérification de la consistance et deuxièmement la créativité pour éviter les conflits. Manipuler une centaine de contraintes est difficile et trouver l'issue requiert de l'ingéniosité. Le concepteur doit dépasser l'espace courant des solutions.

1.4 Intelligence Interactive

Dans une tâche de résolution de problèmes, la synergie entre l'humain et les ordinateurs semble bénéfique en raison de la complémentarité des deux types d'≪intelligence≫. L'humain est avant tout créatif et intuitif alors que la machine possède une mémoire exhaustive, des méthodes de représentation formelle, des possibilités énormes de calcul et une faculté à représenter l'information sous forme visuelle. L'homme et la machine peuvent donc s'enrichir mutuellement et collaborer si l'interaction est bien conçue.

Nous proposons d'établir un modèle basé sur la collaboration entre les humains

[1] routine: 1. savoir ou habileté que l'on acquiert par la pratique plus que par l'étude. 2. conforme à l'habitude prise, qui ne nécessite aucune décision nouvelle.

[2] optimisation multicritère: recherche des valeurs des paramètres qui maximisent la valeur des critères

et les ordinateurs au niveau du raisonnement. C'est une nouvelle approche dans l'interaction humain-ordinateur qui introduit le nouveau concept d'≪intelligence interactive.≫. Nos hypothèses sont les suivantes en ce qui concerne la résolution de problèmes (où le signe > signifie ≪apporte un meilleur résultat que≫):

1. collaboration Humain Machine > Machine.
2. collaboration Humain Machine > Humain.
3. collaboration Humain Machine > travaux de l'Humain + travaux de la Machine (l'opérateur + pris au sens exclusif).

Afin de soutenir la première hypothèse, nous soulignerons combien il est important que l'humain enrichisse le monde de la machine avec la nature (voir 2.3.1), et qu'ainsi la machine puisse traiter la créativité. Pour la deuxième hypothèse, nous décrirons les limites humaines et expliquerons comment la machine peut être un outil permettant d'augmenter les facultés humaines (voir 2.3.2). La troisième hypothèse est plus difficile à aborder. Nous apporterons les éléments qui permettent de la valider. D'abord, nous montrerons que l'interaction est nécessaire. Puis, nous expliquerons la nécessité de la distribution des tâches et du partage des connaissances. Enfin, nous soulignerons l'importance du respect de la complémentarité des intelligences et le rôle déterminant de la visualisation comme espace de travail commun (voir 2.3.3).

1.5 Plan

Dans un premier temps, nous présenterons un bref état de l'art dans le domaine de la conception assistée par ordinateur. Nous montrerons, dans le chapitre suivant, l'intérêt de notre architecture pour soutenir la conception créative. Dans un second temps, nous présenterons notre propre travail et le système Comind: son architecture tout d'abord et la fenêtre principale, ensuite l'assistant Param-Def, le fil d'Ariane du système, puis l'assistant Solve, l'assistant TradeOff et finalement l'assistant Conflict. Nous finirons la présentation des assistants de Comind par l'assistant History et l'assistant ShowCase. Nous présenterons ensuite deux exemples de scénario d'utilisation de notre système avec deux applications différentes: la configuration d'un lotissement d'abord, inspiré de Navinchandra, puis un exemple de robotique parallèle, cher à notre institut. Finalement, nous conclurons et amènerons quelques perspectives.

Chapitre 2

Etat de l'art

2.1 Introduction

Nous présentons dans ce chapitre les travaux importants auxquels nous avons comparé notre approche, desquels nous nous sommes inspirés parfois, et aussi ceux que nous avons cherché à ne pas reproduire.

2.2 Créativité, conception et ordinateurs

L'étude de la conception étant devenue une préoccupation majeure, essentiellement pour des raisons économiques, la conception assistée par ordinateur (CAO) est un pôle phare de la recherche en informatique. On a créé de nombreux outils pour supporter la phase de construction (par exemple, le bien connu AutoCAD). Cependant, la conception créative est une tâche cognitive complexe qui implique des processus de pensée créatifs et qui intervient principalement dans la phase conceptuelle de la création. Or, la phase conceptuelle reste une tâche difficile. Elle requiert une grande attention à cause du grand nombre de critères et de contraintes à manipuler, de la définition floue des objectifs, et du fait des possibles conflits cachés dans la définition. Elle va donc nécessiter la mise en place de compromis. On estime d'ailleurs que 85% des décisions qui permettent d'élaborer un produit, sont prises durant cette phase conceptuelle [Sharp, 1995]. De plus, réduire l'intervalle de temps entre conception et élaboration est la garantie qu'un nouveau concept aboutisse à la réalisation d'un produit qui soit innovant sur le marché.

La conception créative nécessite aussi bien l'étude de la créativité que celle de la conception. Malheureusement, ces processus de pensée sont tous deux mal définis. L'étude de la créativité est abordée dans de nombreux domaines et ses sources sont encore obscures. La créativité est nécessaire dans plusieurs activités comme l'art, la résolution de problèmes, la conception et même les tâches journalières de la vie

de chacun. La conception, quant à elle, est mieux caractérisée. Certaines formes de conception sont formalisées et de nombreuses taxinomies ont été élaborées à son sujet. Cependant, du fait de la croissance de la complexité de notre monde, la conception devient de plus en plus délicate. Tous les jours, nous sommes confrontés à de nouveaux produits issus de la conception et de plus, nous devons parfois nous-mêmes concevoir. Pour ces raisons, des outils aidant les concepteurs sont nécessaires pour augmenter leurs facultés de création et les rendre davantage concernés par la qualité de leur conception.

2.2.1 La créativité

La créativité [1] n'est pas simplement un sujet important en intelligence artificielle et dans le domaine de la conception. C'est un concept-clé pour la compréhension de l'utilisateur et pour la conception des interfaces utilisateur. Les recherches sur la créativité sont multidisciplinaires à cause des différents environnements d'influence auxquels la créativité appartient. On a mieux compris la créativité au fur et à mesure de l'évolution des visions en psychologie. Les plus anciennes théories s'intéressaient à trouver une explication interne aux processus créatifs alors que les dernières théories prennent en compte l'influence de l'environnement ([Lalanne, 1998], [Weisberg, 1993], [Heath, 1993], [Cross, 1989], [Poincaré, 1912a] et [Poincaré, 1912b] pour plus de détails sur l'étude de la créativité). Dans le domaine de la psychologie, une importante question est de savoir si la créativité provient d'un processus de pensée créatif ou si c'est une caractéristique des personnes créatives [Weisberg, 1993]. Sigmund Freud pensait que l'activité créative pouvait être expliquée par la sublimation de forces primitives telles que la sensation d'imperfection ou la compensation d'une perte. Poincaré, quant à lui, constatait une forme de raisonnement différente chez les géomètres et les analystes mathématiciens: les géomètres travaillaient de façon linéaire alors que les analystes raisonnaient suivant un processus plus chaotique [Poincaré, 1912a]. Convergent versus divergent, linéaire versus latéral, et sériel versus holistique sont des caractéristiques qui semblent montrer une dichotomie entre deux formes de pensée [Cross, 1989]. Par ailleurs, Poincaré décomposait la créativité en plusieurs phases: préparation, incubation, illumination, et vérification. La préparation est le travail nécessaire a effectuer sur le sujet de recherche, longue période durant laquelle aucun résultat n'est trouvé. L'incubation est un travail interne. Elle correspond généralement à une période de détente, à des vacances par exemple. Cette étape est inconsciente. L'illumination correspond à l'arrivée brutale et consciente d'une solution.

[1] création: 1. action de donner l'existence, de tirer du néant. 2. l'ensemble des choses créées; le monde, considéré comme créé. 3. action de faire, d'organiser (une chose qui n'existait pas encore. 4. par métonymie, ce qui est créé (par l'homme)

Eurêka! s'est exprimé Archimède dans son bain. La vérification est alors l'étape nécessaire pour valider ou infirmer l'heureuse arrivée [Poincaré, 1912b]. Cependant, la créativité n'est pas simplement à mettre au bénéfice de facultés ou de comportements personnels. Elle dépend aussi de l'environnement social et des opportunités qu'il fournit ([Heath, 1993]). Quand Watson et Crick ont découvert la structure de l'ADN, le rôle de leur environnement de recherche fut primordial. Nous considérons aussi bien les vues introspectives et écologiques à propos de la créativité. Nous croyons qu'il est possible de supporter la créativité et de la stimuler. Cependant, nous ne pensons pas qu'il soit possible de rendre créative des personnes qui ne le sont pas. La machine peut enrichir l'environnement de l'utilisateur et accélérer le processus créatif, cependant elle ne peut pas être elle-même créative.

2.2.2 Créativité et ordinateurs

Le but de beaucoup de chercheurs en IA (Intelligence Artificielle), en conception assistée par ordinateur et en psychologie cognitive a été de pouvoir analyser l'activité de conception, son aspect créatif et scientifique puis d'automatiser ce processus. Leurs recherches nous aident à comprendre comment notre propre cerveau fonctionne pendant l'activité de conception. Elles mettent en lumière la façon dont on peut enseigner aux étudiants les théories et les pratiques en matière de conception et elles offrent aux informaticiens les modèles algorithmiques des processus de conception de routine. Cependant, une grande question n'a pas été élucidée. Savoir si les ordinateurs peuvent être réellement créatifs est une interrogation qui préoccupe toujours les chercheurs. Boden [Boden, 1991] expose les quatre questions de Lovelace permettant d'aborder plus systématiquement le problème de la créativité:

1. les modèles informatiques peuvent-ils nous aider à comprendre comment la créativité humaine est possible?

2. les machines peuvent-elles avoir l'air créatif?

3. les machines peuvent-elles reconnaître la créativité?

4. les machines peuvent-elles être créatives?

Tandis que Boden est principalement intéressée par la première question, les communautés de recherche en Intelligence Artificielle et en conception considèrent que les trois dernières questions sont les plus importantes. La question la plus extrême et la plus provocante est la quatrième. Deux approches principales permettent d'aborder cette question:

- Les réseaux neuronaux et les algorithmes génétiques essayent d'imiter les fonctionnements internes de notre cerveau ou essayent d'utiliser le processus normal d'évolution sur un modèle (respectivement [Coyne et al., 1993] et [Woodbury, 1993]) afin de rendre les machines aptes à produire des conceptions qui semblent créatives [Boden, 1991]. Cette approche est dite sub-symbolique.
- Les techniques basées sur la déduction, la recherche heuristique et les méthodes de raisonnement à base de cas forment un autre champ de recherche dans lequel les scientifiques proposent des mécanismes plus systématiques pour capturer la créativité humaine en terme de règles ou de cas. Cette approche est dite symbolique.

L'une et l'autre des approches ont été comparées à la créativité humaine et jusqu'à présent, ni l'une ni l'autre n'est satisfaisante pour reproduire la créativité. Une telle comparaison, connue plus généralement comme test de Turing, s'inspire d'une hypothèse sérieuse. C'est un essai basé sur la **similitude** plutôt que sur la **différence** entre les capacités de l'homme et celles de la machine.

En informatique, il existe donc deux visions opposées de la créativité: l'une basée sur la similitude et l'autre sur la complémentarité (voir section 2.3, page 11). Certains chercheurs sont impliqués dans la modélisation de la créativité parce qu'ils croient qu'il est possible de rendre les machines créatives. Cela constitue pour nous la première vision concernant la créativité et les ordinateurs. D'autres chercheurs s'intéressent à l'étude approfondie des produits créatifs parce qu'ils pensent que c'est une étape essentielle dans la compréhension de la nature de la créativité [McLaughlin, 1993]. Cette famille de chercheurs se classe à mi-chemin entre les deux visions dont nous faisons le rapport dans ce chapitre.
La créativité et la conception sont une forme d'exploration comme nous l'avons déjà dit. Le caractère ouvert d'une exploration est ainsi en désaccord avec le monde fermé dans lequel tout programme d'une machine se trouve enfermé.

C'est pourquoi d'autres chercheurs, et c'est là la deuxième vision, pensent que la question elle-même n'est pas intéressante. Savoir si les ordinateurs peuvent être créatifs ou non, ne les intéresse guère. Ils remarquent, de plus, que la créativité est non seulement la capacité à résoudre des problèmes, mais qu'elle est aussi la tendance du cerveau à rechercher activement des problèmes, afin d'avoir le plaisir de les résoudre ([Takala, 1993]). Selon ce point de vue, un modèle soulignant les forces de motivation de la créativité peut trouver des moyens de la stimuler.

Il y a donc deux grandes approches complémentaires qui cherchent à traiter de la créativité dans les machines:

- certains chercheurs s'attachent à reproduire la créativité dans la conception (Cf section 2.2.4, page 10): les systèmes à base de règles comme le très fameux système Aaron [Cohen, 1981], les systèmes basés sur le raisonnement analogique [Zhao and Maher, 1992], [Qian and Gero, 1992] (Cf section 2.2.4, page 11), et les systèmes de raisonnement à base de cas [Wills and Kolodner, 1994] (Cf section 2.2.4, page 11).

- d'autres chercheurs essayent de créer des systèmes intelligents qui soient des supports à la créativité (Cf section 2.2.4, page 11): les systèmes d'aide à la conception innovatrice à base de cas ([Gomes et al., 1996], [Navinchandra, 1991], voir section 2.2.4, page 11), les systèmes basés sur les perspectives s'appuyant sur des critiques et explications [Fischer, 1993] (Cf section 2.2.4, page 11), et les systèmes qui utilisent un ensemble d'agents pour faire émerger de nouvelles conceptions [Edmonds et al., 1994].

Nous allons voir dans les différents systèmes d'aide à la conception, l'expression de ces deux visions.

2.2.3 La Conception

La conception est un acte étudié depuis longtemps. Deux mille ans avant J.C. déjà, Hammurabi, roi de Babylone, dictait une loi visant à reconnaître la conception et à se protéger de ses dangers [Gero, 1990]. La conception est une tâche cognitive complexe qui peut être décomposée en plusieurs sous-tâches. [Clavel, 1987] définit la conception comme étant décomposable en de multiples fonctions et, pour lui, la découverte d'une solution correspond au choix d'une solution pour chacune de ces fonctions. John S. Gero a donné la définition la plus populaire de la conception. Selon lui, selon aussi la plupart des chercheurs en intelligence artificielle et conception, il est possible de distinguer deux catégories de conception: la conception routinière[2] et la conception non routinière. Les conceptions routinières sont celles reconnues comme n'étant différentes d'aucune façon des conceptions antécédentes dans leur classe. Plus formellement, dans ce cas, les concepteurs choisissent tous d'utiliser les mêmes variables de conception et en fonction de leur perception de la situation, ils affectent différentes valeurs à ces variables. Les conceptions non-routinières, au contraire, sont différentes des conceptions antécédentes dans leur classe. Dans cette même catégorie de conceptions, on distingue deux sous-catégories, les conceptions innovatrices[3] et les conceptions créatives. Dans les conceptions innovatrices on considère que l'innovation[4], vient de l'usage de valeurs non com-

[2]routinier, ère: qui se conforme à la routine

[3]innovateur, rice: qui tend à innover, fait des innovations

[4]innovation: 1. action d'innover. Innovation par création d'un produit nouveau dans une gamme, etc. 2. chose nouvelles; résultat de l'action d'innover

munes, extérieures à l'ensemble jusqu'alors utilisé, pour les variables de la conception [Gero and Maher, 1993]. Cette définition est un peu trop formelle et elle suppose que les concepteurs pensent toujours en terme de variables. De façon plus intuitive, la conception innovatrice est l'amélioration radicale de la caractéristique d'un produit existant, comme par exemple le passage de la télévision noir et blanc à la télévision couleur. Au cours d'une conception créative, une nouvelle variable de conception est introduite. Nous avons défini une taxinomie différente pour la conception créative (voir 1.3, page 3). Cette définition vient de l'observation du processus de la conception créative. Voyons maintenant les différents systèmes d'aide à la conception créative que nous avons trouvés.

2.2.4 Conception créative assistée par ordinateur

Nous avons donc trouvé deux approches principales dans l'étude de la CCAO (conception créative assistée par ordinateur) , pour plus de détails sur les systèmes existant dans ce domaine, voir le rapport technique [Lalanne, 1998]). Selon notre point de vue, ces deux approches sont dues aux deux paradigmes différents concernant la créativité et des ordinateurs. Un système de conception assisté par ordinateur est une aide à la conception créative s'il permet à l'utilisateur de définir de nouveaux concepts ; il est créatif s'il découvre de nouvelles conceptions par lui-même [Sun and Faltings, 1994].

Dans la première approche, les chercheurs veulent rendre les machines créatives en utilisant des techniques variées. La caractéristique principale des systèmes informatiques qu'ils mettent en oeuvre est qu'ils sont basés sur la connaissance [Gross, 1990]. Dans les systèmes basés sur la connaissance, il y a séparation entre la connaissance et le contrôle; la modélisation symbolique prédomine. La séparation entre la connaissance et le contrôle dans la modélisation de la conception créative s'apparente ≪d'une certaine façon, à la différence entre les produits créatifs et les processus créatifs≫ [Gero and Maher, 1993]. Les produits créatifs impliquent que la connaissance fournisse la base pour la créativité alors que les processus créatifs impliquent que le raisonnement lui-même soit créatif. Par exemple, certains chercheurs utilisent la puissance que possèdent les prototypes comme modèle de calcul sans considérer le rôle que l'exploration joue dans la conception [Loga and Smithers, 1993]. Le rôle de la base de connaissance dans la conception créative est de fournir le contenu et l'organisation de la connaissance de la conception, ceux-ci pouvant être employés pour générer un produit créatif. La construction d'une base de connaissance est liée à plusieurs problématiques telles l'organisation de la mémoire, l'indexation, la recherche flexible, etc. Le raisonnement dans la conception se rapporte à différents modèles de processus tels que la décomposition, la recherche,

l'exploration, l'analogie et la mutation. Une distinction peut être faite au sein de cette approche [Gero and Maher, 1993]:

- selon que la créativité soit contenue dans la base de connaissance ou dans son organisation: la conception créative basée sur les prototypes [Rosenman and Gero, 1993], la conception créative à base de cas ([Riesbeck and Schank, 1989] et [Giretti et al., 1994]), les réseaux neuronaux [Coyne et al., 1993] et les algorithmes génétiques [Woodbury, 1993].

- selon qu'elle provienne du mécanisme de raisonnement utilisé: la conception par combinaison , la conception par mutation [Zhao and Maher, 1992], le raisonnement par analogie ([Qian and Gero, 1992], [Bhatta et al., 1994]) et la conception des premiers principes [Clavel, 1987].

Dans la seconde approche, les chercheurs essayent de trouver les caractéristiques importantes de la créativité humaine afin de l'aider [Fischer, 1993] et les dispositifs qui l'interdisent (influence de la présentation des expériences antérieures dans le processus de conception [Purcell et al., 1994]):

- les systèmes de CAO intelligents ([Lalanne and Pu, 1996], [Clavel, 1987] et [Barbey et al., 1996]), les systèmes à base de cas supportant la créativité ([Gomes et al., 1996], [Burke and Kass, 1994], [Woolf and Hall, 1995], [Pu and Lalanne, 1997] et [Bradley et al., 1994], voir 8.2, page 86), les systèmes à base de critiques et d'explications [Nakakoji et al., 1994], les sytèmes basés sur la résolution de contraintes et l'optimisation ([O'Sullivan, 1998a], [O'Sullivan and Bowen, 1998], [O'Sullivan, 1998b], [Tweedie et al., 1994], [Tweedie et al., 1996] et [Spence et al., 1995]), les systèmes qui reflètent le processus de conception [Chung and Goodwin, 1994], les kits de construction, rôle de l'interaction [Fischer, 1993], les environnements de conception basés sur la connaissance [Fischer, 1993], et les environnements de conception basés sur des agents [Pu, 1994]

2.3 Une nouvelle approche

La séparation que nous venons de faire, entre les systèmes qui essayent de reproduire la créativité et ceux qui tentent d'être son support, n'est pas aussi distincte dans la réalité. Cependant nous cherchons, dans cette thèse, à accentuer les rôles complémentaires de l'humain et de la machine ; nous devons admettre que la créativité humaine et celle de la machine sont différentes. Plus précisément, les hommes et les machines contribuent au processus de conception créative de manière

complémentaire. Nous proposons donc une nouvelle question comme alternative de la quatrième question de Lovelace:

5. la collaboration de l'intelligence humaine et de celle de la machine peut-elle permettre aux humains d'être davantage créatifs?

Nous n'essayons pas de construire des systèmes capables de comprendre l'utilisateur ou de reproduire ses facultés, mais des systèmes capables de compléter ses facultés. L'humain vit dans un environnement stimulant et dynamique qui lui permet d'être créatif et productif. Mais il se trouve face à des limitations cognitives. Nous voulons prendre en compte ses limites afin de libérer son intelligence des contraintes de calcul, soutenir sa perception et son attention, ainsi augmenter ses capacités de raisonnement. Pour cette raison et du fait de la complémentarité de l'humain et de la machine, nous proposons un mode asymétrique d'interaction. C'est en se servant des machines comme esclaves de calcul, de mémoire et de perception, que les humains pourront se concentrer davantage sur des tâches créatives.

Nous posons dans cette section les bases de notre raisonnement qui vont permettre d'aborder notre principe d'≪intelligence interactive≫. Dans un premier temps, nous étudions l'outil informatique, ses spécificités et ses limites. De la même façon, dans un second temps, nous nous intéressons à l'étude du raisonnement humain pendant un processus créatif afin de mieux comprendre ses limites. Ces études nous permettent ainsi d'identifier les critères nécessaires à la réalisation d'un système d'aide pour résoudre des problèmes, dans une démarche où la machine et l'humain partageraient le même but.

2.3.1 Libérer les machines

Les psychologues précisent que les artistes et les scientifiques sont créatifs pendant la période où ils tirent le plus d'inspiration et d'intuition de la nature, de ses objets artificiels, des gens avec qui ils interagissent et des phénomènes normaux auxquels ils sont exposés.

Boden [Boden, 1991], par exemple, décrit comment des scientifiques et des artistes reconnus ont découvert leurs idées dans le trio: bain, lit et bus (≪bath, bed, bus≫). Henri Poincaré ([Poincaré, 1912a], [Poincaré, 1912b]) prenait ≪un bus sur une expédition géologique lorsqu'il tomba soudainement sur une propriété mathématique fondamentale d'une classe de fonctions qu'il avait récemment découverte et qui l'avait préoccupé pendant des jours.≫

Navinchandra [Navinchandra, 1991] utilise le raisonnement à base de cas pour obtenir des conceptions innovatrices. Par exemple, à l'aide de son système, il cherche une solution pour construire une maison dans une pente raide. Une solution créative survient quand le système choisit un cas lié à la construction de maisons en Thaïlande. Le système propose alors d'aplanir la maison à l'aide d'échasses comme bases. Il n'est pas avoué dans ce travail que l'humain joue un rôle important dans la construction, la recherche, et l'adaptation des cas. Il nous semble plus juste de penser que cette solution a été inspirée par l'homme grâce à l'interaction avec la nature (le concepteur revient de vacances en Thaïlande) plutôt que de croire que la base de cas possédait justement ce cas en mémoire et l'a naturellement proposé.

La nature nous a donné l'inspiration nécessaire pour créer des engins volants, l'électricité, de nombreuses oeuvres d'art, etc. Les machines, elles, sont privées de l'inspiration et de l'intuition de la nature. Pour qu'elles soient créatives, elles devraient faire un pas hors de leur monde fermé et commencer à interagir avec la nature. Malheureusement la technologie actuelle dans les domaines concernant la vision, la robotique, et les capteurs n'a pas pu nous permettre de construire des robots et de les envoyer au bord du lac Léman, par exemple, afin de trouver cette inspiration et cette intuition provenant de la nature. En outre, selon Poincaré et Hadamard [Boden, 1991], la créativité consiste en quatre phases: la préparation, l'incubation, l'illumination et la vérification. De ce fait, une expédition de robots dans la nature ne garantirait pas à elle seule le succès de l'illumination. La période de la préparation dans laquelle la personne définit son but créatif et la période d'incubation pendant laquelle le problème créatif est réellement mis de côté manqueront toujours. Notre argument, qui sert de base à notre travail sur l'intelligence interactive, est que nous devons passer par une collaboration entre l'humain et la machine pour pouvoir traiter la créativité dans l'ordinateur. Les humains apporteront leur inspiration et leur intuition provenant de la nature à l'ordinateur. Les machines nous serviront à augmenter nos capacités de raisonnement, comme une voiture augmente nos facultés de locomotion.

Afin de libérer les machines de leur monde pré défini et pour les enrichir de l'interaction avec la nature, nous devons donner aux humains la possibilité d'aider ces machines; de la même façon les humains profiteront de la puissance de calcul des machines.

2.3.2 Un outil pour la cognition humaine

Le paragraphe précédent a souligné la supériorité des humains sur les machines. Nous vivons dans un environnement stimulant et dynamique qui nous permet d'être créatifs et productifs. Mais l'homme rencontre des limites de puissance cognitive et mentale. Par exemple, on sait actuellement que dans la mémoire à court terme, nous pouvons retenir seulement 4 à 7 gros morceaux conceptuels (≪chunks≫) [Card et al., 1983]. Nous avons souvent du mal à visualiser un ensemble de vecteurs dans trois dimensions, c'est encore plus difficile dans un nombre de dimensions plus élevé. Avec la croissance rapide des technologies de l'industrie informatique, nous avons pu utiliser les machines en tant qu'outil cognitif et outil de calcul. Nous comptons sur leur puissance pour nous aider à accomplir des tâches difficiles.

2.3.3 Hypothèse: collaboration basée sur la complémentarité

Nous avons repris les concepts énoncés précédemment afin d'en extraire les critères nécessaires pour définir notre modèle. Nous proposons d'établir un modèle basé sur la collaboration entre les humains et les ordinateurs au niveau du raisonnement. C'est une nouvelle approche dans l'interaction humain-ordinateur qui introduit le nouveau concept d'≪intelligence interactive.≫.

2.3.3.1 Collaboration vs Coopération: interaction

La connaissance distribuée n'est pas seulement $1 + 1 = 2$. Ce n'est pas simplement la division d'une tâche en sous-tâches secondaires réparties entre des personnes différentes travaillant dans des pièces différentes, sous-tâches assemblées quand les personnes ont terminé leur travail. Ce n'est pas simplement une coopération[5]. La connaissance distribuée est quelque chose de plus. L'idée est que le tout est plus grand que la somme des parties: $1 + 1 > 2$. C'est la distinction de base entre la coopération et la collaboration[6]. Dans une collaboration, les gens travaillent ensemble, sans se répartir le travail en sous-tâches. Néanmoins, il y a une certaine division du travail. La différence principale est que pendant la collaboration, il y a interaction. C'est l'interaction qui apporte la synergie et qui permet $1+1 > 2$. C'est la qualité de l'interaction qui est la garantie d'une bonne collaboration. Cependant, la qualité de l'interaction est difficile à mesurer. Nous montrons dans le paragraphe suivant que le partage de la connaissance est une condition nécessaire à la réussite de la collaboration. Nous montrons ensuite l'importance de la complémentarité.

[5]Coopération: 1435 ; ≪cooperatio≫, du latin ≪part prise à un travail conjointement fait.≫.
[6]Collaboration: 1829 ; ≪travail en couple≫, 1753; du latin ≪collaborare≫. - collaborer.

Notre approche: architecture flexible

Notre volonté est de permettre à l'utilisateur de collaborer avec la machine lorsqu'il le désire et à n'importe quelle étape de la conception. Pour cette raison, la machine ne va pas imposer une séquence d'utilisation. L'architecture ouverte que nous proposons et que nous présentons dans le chapitre suivant permet de libérer les machines de leur monde pré défini et de les enrichir des interactions avec la nature, parce qu'elle permet aux humains d'exprimer leur créativité et leur intuition. Nous devons considérer un certain nombre de paramètres dans la conception de telles systèmes: l'ergonomie du système, son architecture et l'organisation sociale entre l'homme et la machine comme collaborateurs et co-contribuants. Le mécanisme principal de cette collaboration est basée sur une boîte à outils composée d'assistants logiciels aux services de l'utilisateur. Les sociologues montrent que l'interaction entre l'homme et la machine est principalement sociale [Nass et al., 1994]. En outre, l'anthropomorphisme apporté par des assistants d'inspiration humaine est supérieur à d'autres métaphores telles que la programmation orientée objet. Enfin, le sens commun nous indique qu'il est beaucoup plus facile de dire à son assistant ce que l'on a vu dans le métro en venant travailler qu'à son programme nommé inspiration.lisp. La section 3.1 (page 19) explique très précisément les motivations de ce choix et pourquoi il permet aux connaissances d'être distribuées autour de l'utilisateur. Cette architecture permet d'aider les machines, grâce à la collaboration possible dans une démarche de créativité humaine, comme elle permet d'aider les humains qui bénéficient de la puissance de calcul des machines.

2.3.3.2 Partage des connaissances

Dans l'étude de la collaboration, l'efficacité vient de la capacité à établir un système cognitif commun, c'est-à-dire à instaurer une vision et une connaissance partagées du problème. Pour cette raison, certains parlent de connaissance partagée au lieu de connaissance distribuée lorsqu'ils cherchent à trouver le meilleur environnement pour la collaboration [Dillenbourg et al., 1996]. Bien que cette observation semble valable pour la collaboration entre humains, il est difficile de parler de connaissance partagée dans la collaboration humain-ordinateur du fait du manque de conscience des machines. Cependant, il est raisonnable de penser que si machines et humains travaillent sur les mêmes informations, qu'ils se doivent de passer l'un par l'autre pour avancer dans leur raisonnement, alors le partage de la connaissance sera garanti.

Notre approche: espace du problème et des solutions partagées

L'environnement de conception, que nous proposons, permet aux humains et aux ordinateurs de collaborer et de coopérer, partageant leurs ressources cognitives et de calcul. Le partage des connaissances est une caractéristique importante dans notre système. La machine et l'utilisateur partagent la même vision des connaissances, le même état du problème. A tout instant, la machine et l'utilisateur peuvent intervenir sur l'espace du problème, sur sa définition ainsi que sur l'espace des solutions. Nous verrons dans le chapitre sur l'assistant d'aide à la définition du problème qu'il sert de fil d'Ariane dans le processus de conception et qu'il unifie ainsi tous les assistants et l'utilisateur.

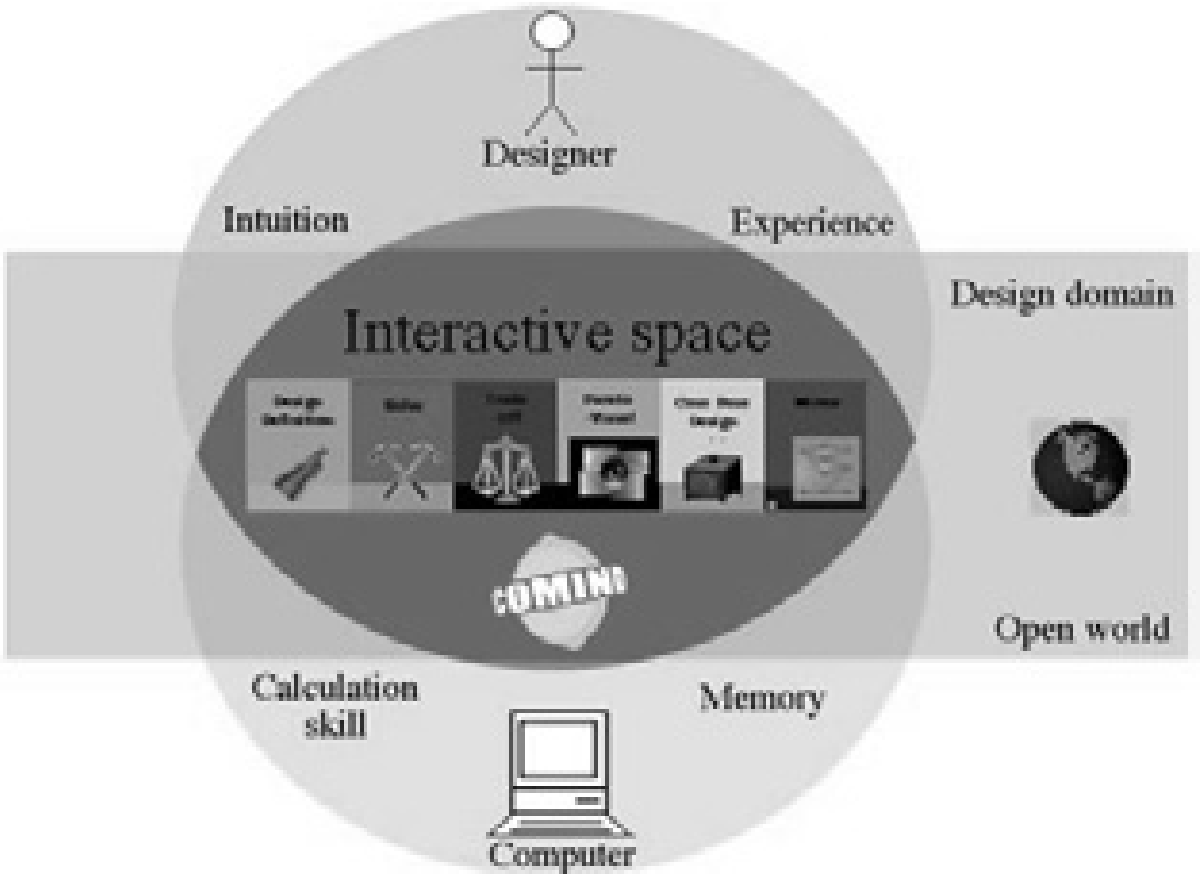

Figure 2.1: L'architecture basée sur la complémentarité de Comind

2.3.3.3 Complémentarité des facultés et visualisation

Nous montrons dans cette thèse que la visualisation permet à l'homme et à la machine d'agir l'un sur l'autre. En effet, la visualisation est la voie par laquelle nous, humains, avons appris la plupart des abstractions et grâce à laquelle nous résumons

graphiquement des longues séquences de chiffres. La visualisation est aussi un média très expressif qui permet plusieurs niveaux d'abstraction et qui permet de communiquer rapidement et efficacement. De plus, les machines sont douées pour représenter graphiquement l'information.

Cependant, une bonne interaction n'est pas la seule condition nécessaire pour réaliser une bonne collaboration dans une tâche de résolution de problèmes. Une bonne collaboration dépend aussi, dans l'interaction humain-ordinateur, de la qualité de la distribution des tâches. Les machines sont douées pour certains types de travaux et les humains pour d'autres.

Notre approche: visualisation pour satisfaire l'interaction et augmenter l'espace de raisonnement

Nous avons fait quelques études afin d'identifier l'ensemble des activités de conception qui conviennent davantage aux humains et celles qui conviennent mieux aux machines. Nous avons pu ainsi définir différents assistants logiciels qui complètent les facultés humaines. Ils vont permettre d'améliorer les facultés de **calcul**, d'**attention**, de **perception** et de **mémoire**. La visualisation permet au concepteur de raisonner sur un espace plus large et de se focaliser sur un espace sans perdre la direction initiale. La visualisation se comporte comme un outil d'exploration qui augmente les facultés de raisonnement du concepteur en agrandissant son espace de travail. Elle permet d'augmenter la compétence de l'utilisateur dans l'évaluation de l'espace et de réduire ainsi l'intervalle entre les objectifs et la définition (point clé des bons systèmes d'aide à la conception créative, voir [Lalanne, 1998]). Elle fournit un espace interactif pour l'utilisateur et la machine, leur permettant ainsi de communiquer. Nous verrons alors que la visualisation va aider le concepteur à découvrir plus facilement les conflits dans un problème sur-contraint et qu'elle l'aidera aussi à faire des compromis dans un problème sous-contraint. Nous montrerons que la visualisation augmente la perception de l'utilisateur et que profitant de l'aide de la machine au niveau du calcul et du raisonnement, elle va être un terrain propice à la créativité.

2.4 Conclusion

La créativité est un processus qui requiert le dépassement des règles établies. Les personnes créatives explorent de nouveaux espaces et élargissent la vision établie de leur domaine. Afin de formaliser la conception, la représentation courante consiste à considérer le vocabulaire et les paramètres comme les variables du problème.

Trouver une solution consiste alors à fournir un ensemble de valeurs (une valuation) qui satisfasse aussi bien les contraintes que les objectifs pour chaque variable du problème. Cependant, les problèmes sur et sous-contraints étant difficiles à résoudre, il sera souvent difficile de trouver une solution qui satisfasse toutes les contraintes et objectifs. Le concepteur doit faire des compromis (≪trade-off≫) ou modifier au mieux son problème afin d'échapper aux conflits (≪break-through≫). Parfois, le concepteur se trouve dans une impasse.

Selon des recherches actuelles, la créativité consiste en la découverte de nouvelles variables, variables qui permettent d'éviter les conflits entre contraintes ou objectifs. Cette opération est souvent nommée une mutation. Une autre possibilité est de permettre au concepteur de se balader plus vite dans l'espace des solutions, avec une meilleure perception et une plus grande mémoire. C'est ce que nous proposons dans cette thèse. La visualisation est particulièrement importante car elle fournit un espace interactif qui permet à la machine et à l'humain de communiquer et ainsi de bénéficier de chacune de leurs facultés respectives. Elle augmente aussi nos facultés visuelles et perceptives. Il y a donc deux processus cognitifs importants dans la conception créative:

- faire des compromis, comparer (≪trade off≫)
- relaxer, reformuler, élargir (≪break-through≫).

Comme nous l'avons vu, la créativité est difficile à reproduire dans une machine parce que les machines ne sont pas capables de violer des règles, de les transcender. Cependant, il est possible de ne pas interdire aux machines de traiter de la créativité grâce à la visualisation et à l'interaction.

Chapitre 3

Architecture

Nous présentons, dans ce chapitre, l'architecture du modèle introduit dans les chapitres précédents. COMIND (pour **CO**mputer aided creativity and **M**ulticriteria optimization **IN D**esign) est le nom du système que nous avons développé qui permet à des humains et à des ordinateurs de collaborer et de coopérer, partageant leurs ressources cognitives et de calcul.

3.1 Système de tâche

Nous avons fait des études afin d'identifier l'ensemble des activités de conception qui conviennent mieux aux humains et celles qui sont caractéristiques des machines [Pu and Lalanne, 1996b]. Les activités ne sont pas hiérarchisées dans notre système pour laisser toute liberté à l'utilisateur. Elles sont distribuées sous forme de tâches. L'utilisateur est ainsi libre de choisir sa propre voie d'interaction. Notre approche intègre plusieurs systèmes intelligents dans une architecture ouverte et distribuée. Cette architecture facilite la collaboration homme machine, la coexistence des créativités humaine et artificielle ainsi que leur co-contribution. Une architecture ouverte est différente de la nature traditionnelle algorithmique des logiciels; elle organise ses fonctionnalités au niveau de la distribution de tâches plutôt qu'au niveau de la performance de tâches.

Plus spécifiquement, nous proposons d'utiliser la méthode des assistants logiciels pour concevoir et développer une telle architecture. Des assistants automatiques ou semi-automatiques collaborent avec les concepteurs humains, ils peuvent coopérer les uns avec les autres afin d'atteindre un objectif commun. Notre environnement de conception, centré autour de l'humain, offre de nombreux modules fonctionnels sous forme d'assistants, c'est-à-dire des agents de service. L'utilisateur peut choisir d'utiliser l'assistant qu'il désire à tout moment, comme s'il disposait d'une boîte à outils d'aide au raisonnement. Il est important que la connaissance soit distribuée

car elle existe dans le monde et dans notre tête. L'utilisabilité d'un outil dépend de la connaissance qu'il apporte sur lui-même dnas son environnement [Norman, 1988]. Si la connaissance est insuffisante, il est totalement impossible d'utiliser l'outil. Par exemple, la connaissance que l'environnement nous donne et celle construite dans notre tête sont toutes deux essentielles dans notre fonctionnement quotidien. Nous voulons montrer le lien solide entre les environnements centrés autour de l'humain et les environnements de conception basés sur la métaphore des assistants. Notre objectif est de pouvoir s'échapper du monde fermé informatique grâce aux assistants utilisateurs. Un environnement centré humain est basé sur le concept de la connaissance distribuée. Selon cette théorie, la connaissance est située dans les objets que l'on désire utiliser (≪situated cognition≫). Une poignée de porte doit indiquer comment elle s'utilise. Par exemple, je n'arrive jamais à ouvrir du premier coup les portes de mon école. C'est parce qu'elles ne fournissent pas la connaissance nécessaire pour que je les ouvre. Afin d'utiliser au mieux un outil, la connaissance doit être véhiculée par l'outil lui-même. Les assistants utilisateurs sont capables d'indiquer leurs fonctionnalités, de plus ils sont capables d'interagir avec l'utilisateur. C'est pour cela que nous les nommons assistants *utilisateurs* . Ces assistants apparaissent sous forme d'icônes dans l'espace de travail et ils sont divisés principalement en deux familles: les **assistants de calcul** et **les assistants réflexifs**. En résumé, en fournissant des outils intelligents et réflexifs qui complètent des qualifications humaines et en rendant les systèmes logiciels plus transparents et interactifs, les utilisateurs ont plus de contrôle sur les décisions prises et peuvent ainsi explorer leur créativité plus librement.

3.1.1 Architecture triangulaire

Les assistants de calcul et l'utilisateur humain doivent fonctionner en synergie dans tout le processus de la résolution de problèmes. Afin de réaliser une coopération pertinente, un mécanisme efficace de transmission doit être mis en place. Notre expérience nous a prouvé que la visualisation interactive peut être le candidat parfait pour satisfaire cet objectif. Les machines sont puissantes pour leurs qualifications de calcul et les humains sont particulièrement doués pour leurs qualités intuitives et créatives. Afin de les rendre capables de coopérer, nous employons une architecture triangulaire contenant l'utilisateur, les assistants de calcul et les assistants réflexifs (figure 3.1) connectant les outils entre eux. Les outils d'interfaçages sont par extension considérés comme étant des assistants réflexifs. Ils reflètent le travail de l'humain et des assistants de calcul. Chaque élément peut communiquer l'un avec l'autre. Les outils d'interface servent de zone interactive de travail permettant la coopération entre l'homme et la machine. L'utilisateur peut utiliser un même assistant réflexif (outil d'interfaçage) afin de communiquer avec plusieurs assistants

de calcul ou utiliser un même outil de calcul avec différents outils d'interfaçage afin d'avoir différents points de vue d'un même calcul. En outre, des interactions simples entre les assistants de calcul et les utilisateurs peuvent avoir lieu d'une façon directe, sans aucun intermédiaire. Cependant, des interactions complexes nécessitent la présence d'outils de visualisation interactifs spéciaux. Cette indépendance entre les acteurs permet d'avoir une architecture ouverte, un genre de modularité permettant toute les combinaisons d'outils. La visualisation interactive agit principalement en tant que lien bi-directionnel entre l'intelligence artificielle et l'intelligence humaine. Le bénéficiaire de cette transmission est le processus de résolution des problèmes.

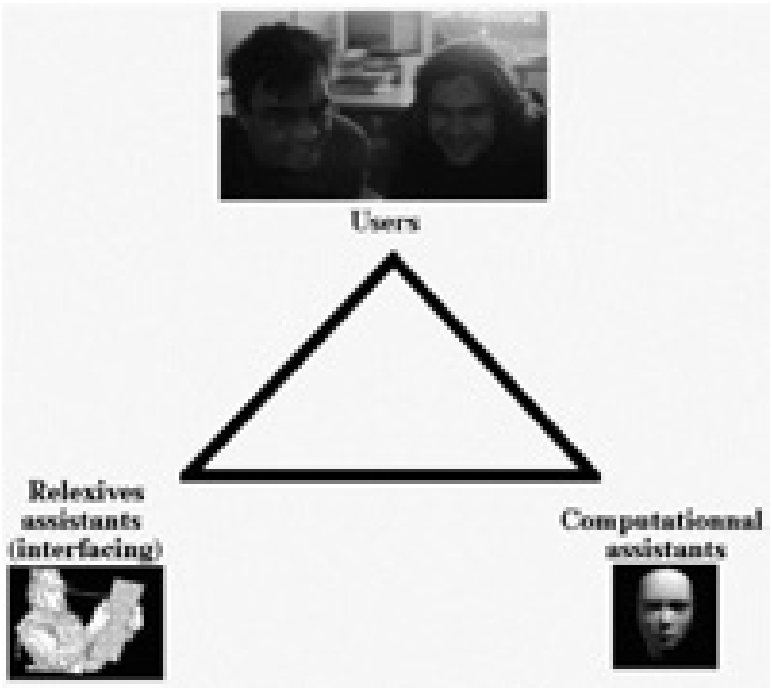

Figure 3.1: L'architecture triangulaire de Comind

3.1.2 A chacun son rôle

Chacun trouve donc son rôle dans cet architecture. Les rôles ne sont pas figés et chaque acteur peut effectuer une tâche pour laquelle il n'est pas spécialiste. Cependant, nous pouvons caractériser les spécialités de chacun des acteurs.

- Les humains créent, dirigent et supervisent.
- Les **assistants de calcul** s'occupent de vérifier les inconsistances dans l'espace de conception, de trouver des solutions partielles et complètes, de présenter des exemples de conception, d'aider l'utilisateur à visualiser un ensemble de bonnes solutions et de détecter des conflits entre contraintes. Ces assistants

sont semi-automatiques; ils ne sont appelés que sur demande de l'utilisateur. Ces assistants pourraient devenir davantage autonomes à partir du moment où une certaine confiance a été établie entre l'utilisateur et l'assistant. Cependant, nous voulons minimiser la communication entre assistants afin de simplifier les protocoles utilisés dans cette architecture.

- Les **assistants réflexifs** sont eux entièrement autonomes et leur but principal est de servir de modèle cognitif à l'utilisateur. L'assistant History montrera par exemple que le concepteur a juste fait une session de *Brainstorming* et qu'il est maintenant en train de modifier les paramètres de la conception. Il serviront, en général, d'outil de visualisation du travail des assistants de calcul ou de l'utilisateur.

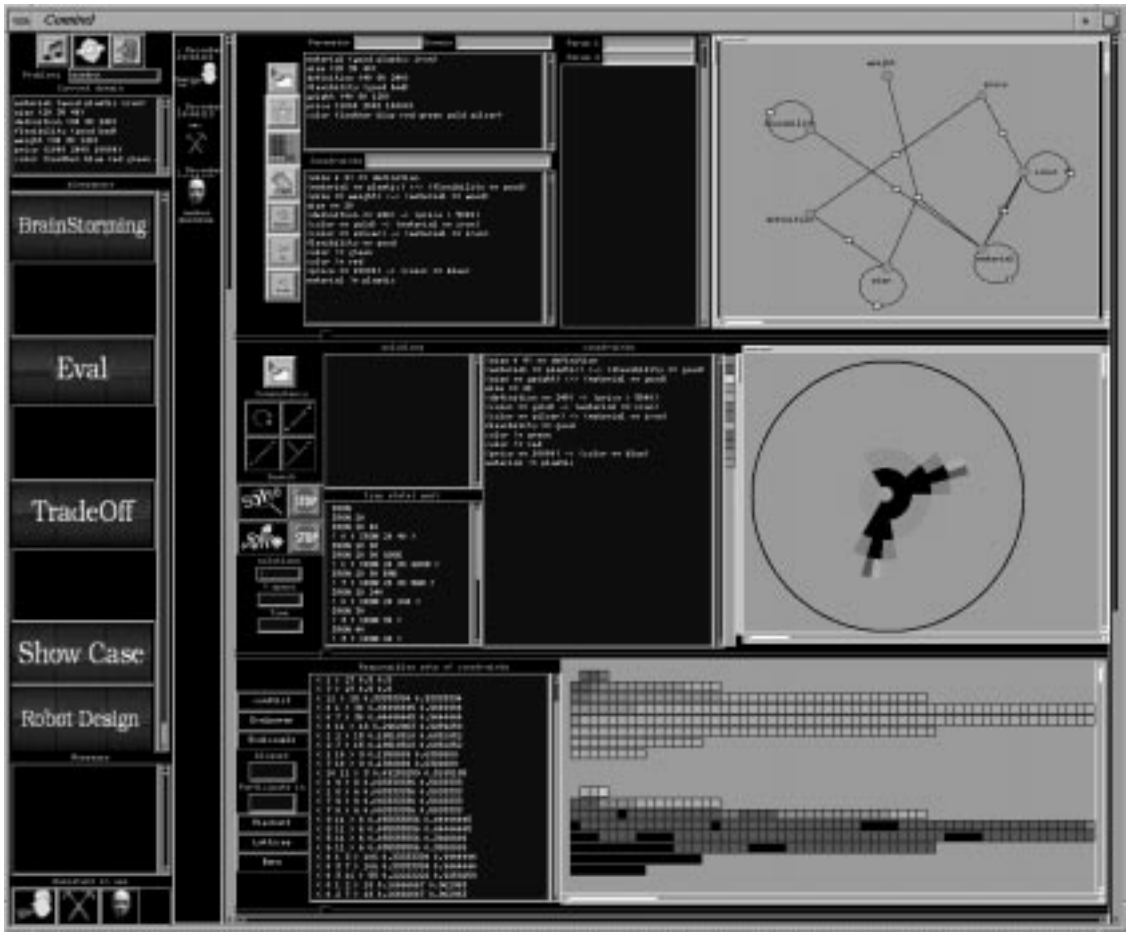

Figure 3.2: L'interface principale de Comind

La figure 3.2 montre l'interface de COMIND ainsi que ses principaux systèmes intelligents. En haut à gauche de l'interface, les assistants disponibles sont présentés. Ils sont dans l'espace de repos. L'espace de travail est juste en dessous. L'utilisateur

peut sélectionner un assistant et le faire glisser vers l'atelier de travail. Il s'ouvre alors dans la fenêtre de droite où plusieurs assistants peuvent être ouverts à la fois. Entre les deux, l'assistant History, un des assistants réflexifs, conserve la séquence d'utilisation du système. Enfin, en bas à gauche de l'interface, l'état du problème courant est représenté.

3.1.3 Tout le monde y trouve son compte

Les compétences et connaissances étant distribuées sous différentes entités, tous les types d'utilisateurs trouveront de l'assistance. Par exemple, nous avons imaginé trois types d'utilisateur:

- un utilisateur précis qui possède des idées a priori: Il possède des contraintes précises qu'il veut satisfaire (par exemple, dans l'aménagement d'une cuisine: des contraintes topographiques comme le plan de la cuisine, les meubles à placer, etc.). L'utilisation de l'aide à la satisfaction de contraintes sera particulièrement importante pour lui.

- un utilisateur flou qui possède des critères flous (tels que la clarté, l'ancienneté, le style, etc pour l'aménagement d'une cuisine.). L'utilisation de l'aide à l'optimisation multi-critères lui sera d'une grande utilité.

- un utilisateur indécis qui n'a aucune idée a priori. L'utilisation d'un catalogue de solutions (assistant réflexif *Showcase*) sera alors utile pour l'aider à définir ses critères et contraintes.

3.1.4 Sous l'angle des contraintes et des critères

Contraintes et critères sont définis à un niveau global et seront partagés par tous les sous-systèmes intelligents correspondant aux tâches de la conception. Un système en tâches hiérarchiques ne pose pas de problèmes supplémentaires pour la prise en compte des critères et des contraintes. Néanmoins, il contredit le déroulement ≪ouvert≫ d'une tâche de conception. Les assistants qui nous intéressent tout particulièrement concernent la recherche de solutions par résolution de contraintes et la recherche de compromis par optimisation multi-critères. Notons, de plus, que la résolution des conflits peut être une étape particulièrement importante dans la recherche de la meilleure solution.

Il n'est pas évident que la phase de relaxation des conflits et la phase d'optimisation multi-critères dussent être séparées parce que des compromis doivent souvent être faits au niveau de la résolution de problèmes sur-contraints, de même que des conflits doivent être relaxés dans l'optimisation. Nous avons néanmoins choisi cette

séparation pour une plus grande clarté car nous pensons que ces deux tâches sont des processus cognitifs différents chez l'utilisateur. De plus, du fait du caractère distribué et ouvert de l'architecture de notre système Comind, l'utilisateur est libre d'utiliser les assistants *ElicitConflict* (pour la résolution de problèmes sur-contraints) et *TradeOff* (pour l'optimisation multicritères) en parallèle, il peut donc les utiliser de façon synergique et les faire coopérer.

3.2 Assistants réflexifs

Les assistants réflexifs servent, en général, à refléter le travail de l'humain, des assistants de calcul ou des deux ensemble. Il y a une multitude d'assistants réflexifs attachés à chaque tâche de conception; ils servent d'espace interactif de travail entre l'humain et la machine. Il y a deux gros assistants réflexifs qui servent le processus de conception global. Nous présenterons plus en détail ces assistants dans le chapitre 8, page 81:

- l'assistant **History** est la mémoire du processus de conception. Une diminution d'attention dans un processus décisionnel peut être catastrophique; le concepteur va manquer de bonnes solutions ou manquer de solutions alternatives dans son raisonnement. Les cartes cognitives [Boden, 1991] sont utiles pour guider des humains dans l'exploration de territoires inconnus. Puisqu'il est dangereux pour les concepteurs de ne pas glisser dans leurs voies habituelles de pensée, les cartes cognitives peuvent servir d'outil supportant le système attentionnel humain; ainsi les concepteurs pourront explorer plus librement leur pensée sans se perdre ou choisir de la changer. C'est un contrôle supplémentaire qui donne plus de liberté au concepteur. En outre, refléter la recherche du concepteur peut lui donner le sentiment que l'espace de recherche est fermé et ainsi le rassurer ou lui donner plus envie de le dépasser ou de l'agrandir, par la création.

- l'assistant **ShowCase** ([Pu and Lalanne, 1997]) permet, dans le domaine de la conception, d'organiser des documents multimédia, de les récupérer et d'interagir avec eux. Son rôle est aussi bien de guider un novice que d'inspirer un expert à partir d'une base de cas d'exemples.

Ces assistants sont dits ≪réflexifs[1]≫: Il ne faut pas les appeler ≪réflectifs[2]≫ parce qu'ils ne modifient pas eux-mêmes directement la définition de la conception mais ils

[1]REFLEXIF, IVE [Refléksif, iv] adj. 1. Phys. Vx. Qui se réfléchit, fait réflexion sur... 2. (1612). Philos. Propre à la réflexion, au retour de la pensée, de la conscience sur elle-même. 3. (XXe). Math. Relation réflexive: relation binaire dans un ensemble, telle que tout élément de cet ensemble est en relation avec lui-même.

[2]REFLECTIF, IVE [Refléktif, iv] adj. Philos. Qui résulte de la réflexion, de la méditation. Dispositions réflectives. Idées réflectives. - REM. Réflectif ≪tourné vers≫ s'entend de ce qui

aident à réfléchir, à prendre du recul par rapport au processus en cours, à organiser et à donner un feedback du processus continu de la conception. De cette manière, ils aident les concepteurs à réfléchir sur leur propre conception et donc indirectement à l'améliorer.

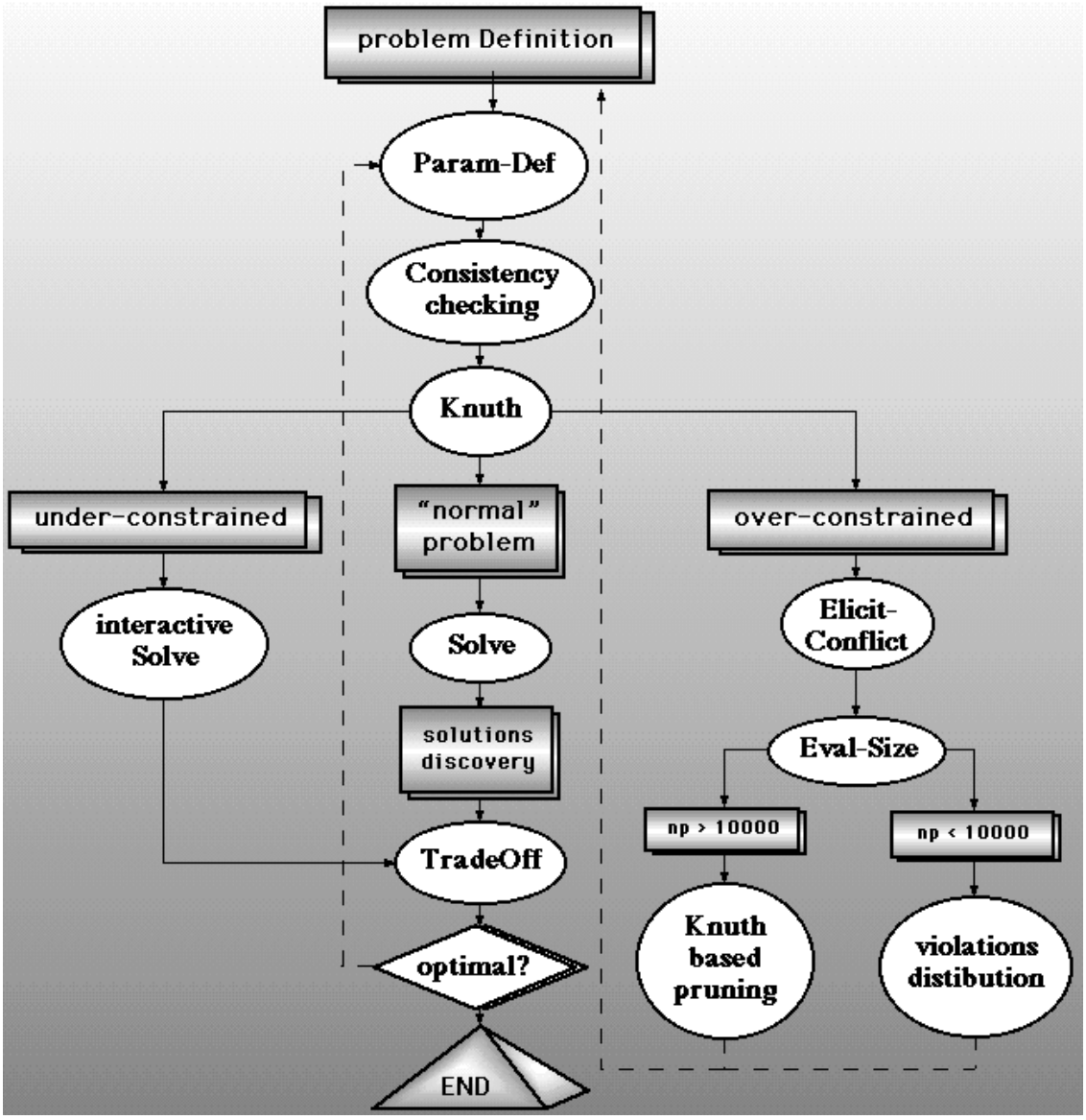

Figure 3.3: Un exemple de distribution des tâches dans Comind

est issu de la réflexion et va produire ses effets en tant que tel, alors que réflexif ≪réfléchi sur≫ s'entend de la réflexion elle-même en tant que moyen d'investigation de la pensée. - Psychol. Facultés réflectives, ≪se dit dans le système phrénologique de Spurzheim, de la comparaison et de la causalité (Bescherelle, Dict.).

3.3 Les assistants de calcul

En contraste avec les assistants réflexifs, les assistants de calcul accomplissent réellement des tâches pour les concepteurs humains. Leur rôle principal est de porter certains des fardeaux de calcul de l'humain et de fournir des visualisations (sous-assistants réflexifs) pour faciliter la perception de l'humain dans des espaces dimensionnels élevés. La figure 3.3 est un exemple d'utilisation de Comind; chaque cercle représente l'utilisation d'un assistant de calcul.

- Afin de définir un problème, le concepteur peut utiliser l'assistant **Brainstorming**. Dans cette étape, le concepteur peut introduire librement ses idées au sujet du problème dans un éditeur. L'assistant *Brainstorming* propose des analogies avec des sessions plus anciennes. Le but est de produire autant d'idées que possible. D'autres aides sont fournies afin de supporter la structuration du problème. Le but de cet assistant est de garder une trace des idées et des buts initiaux, et de supporter la formalisation du problème.

- Après la production d'un bon nombre d'idées, le concepteur peut utiliser l'assistant **Param-Def** afin de définir le problème d'une façon plus formelle et de pouvoir le résoudre. La définition suit la structure d'un problème de satisfaction de contraintes. Cependant, les contraintes peuvent être définies par des règles ou des matrices de logique. Les règles d'écriture peuvent être une tâche difficile pour un concepteur. L'assistant *Param-Def* supporte ce processus en fournissant à l'utilisateur un menu contenant tous les types possibles de règles et leurs traductions intuitives. Il fournit également une visualisation du graphique créé par des paramètres et des contraintes. C'est un bon feedback visuel pour l'utilisateur parce qu'il donne une vue d'ensemble du problème comme réseau de contraintes.

- Puisque les problèmes peuvent être de natures différentes, nous avons dû créer des outils capables de traiter chacune de ces familles. La figure 3.3 résume les différentes étapes généralement empruntées dans l'utilisation de Comind. Notre motivation principale dans le découpage des tâches venait de notre croyance que la créativité dans la conception vient de deux types de processus cognitifs: dans le premier, il faut faire des compromis alors que dans l'autre, il faut résoudre des conflits. Ces deux types de processus se trouvent liés à une autre dichotomie. Il y a deux types extrêmes de problèmes: sur et sous-contraints. Dans le cas sur-contraint, l'espace de solution est clairsemé et souvent nul alors que dans le second cas, il y a beaucoup trop de solutions.
  L'utilisateur peut évaluer le nombre de solutions que possède son problème afin de savoir à quelle famille de problèmes il fait face et quels outils sont les

mieux adaptés pour l'aider. Il utilise l'algorithme de Knuth que nous décrivons plus loin qui permet d'obtenir une évaluation rapide du nombre de solutions.

- Dans les problèmes sous-contraints, trop de solutions sont possibles. Il est difficile d'évaluer les solutions optimales parce qu'elles peuvent être concurrentes, ou Pareto optimales [Pareto, 1896]. Il faut faire des compromis pour trouver la solution optimale. Cependant, lorsque le nombre de critères est grand, les choix sont difficiles. L'assistant **TradeOff** fournit un éditeur pour définir les critères de la conception. Le calcul de ces critères est une évaluation qui intervient après la recherche. C'est un post-traitement. Les critères sont calculés pour toutes les solutions trouvées pendant le processus de recherche. Nous proposons plusieurs visualisations de cette étape. Ainsi, le concepteur peut évaluer les solutions selon les critères définis dans l'éditeur. Il est ainsi également possible d'évaluer la qualité et la précision des critères. Si l'utilisateur n'est pas satisfait par les solutions trouvées selon cet ensemble de critères, il peut abandonner un voisinage et opter pour un autre dans l'espace de recherche. Bien qu'il semble que nous permettions au concepteur ≪de tirer dans l'obscurité≫ pour de découvrir des conceptions créatives, ce genre de concept fait partie de la découverte. Nous montrerons que ce type de problème, appelé l'optimisation multi-critères, peut être mieux traité à l'aide des machines, celles-ci fournissant de bons outils de visualisation et augmentant la perception de l'utilisateur (voir 6).
- Dans les problèmes sur-contraints, il n'y a aucune solution. Le concepteur doit ajouter des paramètres ou relaxer des contraintes afin de découvrir des solutions. Ce processus, appelé la mutation, est central dans beaucoup de problèmes de conception où la créativité des humains est exigée. Afin de découvrir les conflits et de les résoudre, deux grandes facultés sont donc nécessaires :
  - la consistance pour contrôler la validité du problème
  - la créativité pour agrandir l'espace des solutions, le modifier, pour éviter ainsi les conflits et évoluer vers un nouvel espace.

  Ces deux facultés correspondent justement aux qualités respectives de la machine et de l'humain. En effet, la limitation de la mémoire humaine est un problème. Traiter une centaine de contraintes est difficile et l'évasion exige de l'ingéniosité; le concepteur doit dépasser l'espace actuel des solutions. Cette étape est particulièrement importante dans le processus concepteur parce qu'éviter un conflit est souvent fait par

mutation; afin de s'échapper d'un conflit, l'utilisateur doit reformuler le problème ou relaxer certaines contraintes. L'assistant **ElicitConflict** est basé sur une visualisation des règles contradictoires. Il indique les parties du réseau de contraintes qui sont en conflit. Le concepteur peut alors choisir une zone en conflit et voir les valeurs interdites par les règles correspondantes. Le point important dans cette étape est d'aider l'utilisateur à jouer avec les contraintes afin de résoudre les conflits qui interdisent la découverte des solutions. L'interactivité visuelle est également importante parce qu'elle permet, en interagissant avec les visualisations, une meilleure appréciation de l'influence des contraintes les unes sur les autres.

– Dans le cas où le problème n'est ni sous-contraint ni sur-contraint, il faut résoudre le problème. Cependant, lorsque le nombre de paramètres et de contraintes augmente, la résolution du problème devient impossible pour un solutionneur humain. Heureusement, les techniques de l'intelligence artificielle offre de nombreux algorithmes de recherche pour la résolution de problèmes sous forme de contraintes. Notre assistant **Solve** utilise ces algorithmes de recherche pour trouver automatiquement des solutions de conception. Il fournit également différentes visualisations de l'espace de recherche. Nous avons implanté différents algorithmes pour résoudre des problèmes par satisfaction de contraintes. Nous fournissons à l'utilisateur différentes visualisations du cheminement de ces algorithmes. L'utilisateur peut ainsi contrôler ses recherches, les arrêter lorsqu'il le souhaite et même faire coopérer différents types d'algorithmes.

3.4 Conclusion

Notre hypothèse principale est que l'interaction entre l'homme et la machine joue un rôle important dans la conception informatisée, particulièrement quand la créativité est nécessaire. En fournissant plusieurs systèmes intelligents et réflexifs qui complètent les facultés humaines, dans une architecture ouverte et distribuée, en les rendant plus transparents et interactifs, les utilisateurs ont plus de contrôle sur les décisions, ils peuvent ainsi faire appel à leur créativité plus librement.

Chapitre 4

Définir le problème

≪Définir un problème, c'est à moitié le résoudre≫.

4.1 Introduction

Ce dicton est souvent utilisé afin de signifier l'importance de la définition d'un problème pour bien pouvoir le résoudre. De plus, selon notre expérience, il est difficile de séparer la définition d'un problème du reste du processus de résolution. Le concepteur modifie sa définition tout au long de sa conception, jusqu'à ce que des solutions satisfaisantes soient atteintes. Pour cette raison, nous avons choisi d'utiliser l'assistant *Param-Def* comme le fil d'Ariane de notre système, qui réunit tous les autres assistants. L'utilisateur est en effet souvent amené à utiliser cet assistant avec d'autres assistants. D'une certaine façon, l'assistant *Param-Def* est la mémoire de travail du processus de résolution de problèmes. Nous décrivons en détail dans ce chapitre l'assistant *Param-Def*. Tout d'abord, nous présentons en quelques lignes l'assistant *Brainstorming* qui sert d'introduction à l'assistant *Param-Def*.

4.2 L'assistant Brainstorming

Cet assistant a été implémenté dans le but d'aider l'utilisateur à définir son problème de la façon la plus naturelle possible et de soutenir l'étape de génération d'idées en proposant des analogies. Un objectif important dans cette étape était aussi d'amener le concepteur vers une structuration de ses idées, le problème pouvant ainsi être formalisé et résolu automatiquement. Dans un premier temps, l'utilisateur peut exprimer librement son problème et ses idées dans l'éditeur de l'assistant en utilisant son langage naturel. L'assistant lui propose des analogies avec des anciennes sessions. L'objectif premier est de générer autant d'idées que possible. Nous pourrions imaginer, dans le future, une collaboration avec des assistants cherchant à travers

l'internet des analogies. Dans un second temps, d'autres outils sont proposés au concepteur afin de l'aider à structurer le problème. Il peut ainsi graphiquement créer un réseau de relations entre ses idées (voir figure 9.2, page 95). Le rôle de l'assistant Brainstorming est donc de supporter l'externalisation de la définition du problème, d'amener le concepteur vers une structuration de ses idées et aussi de garder une trace de ses idées initiales afin qu'il puisse s'en libérer. Suivant cette même idée de structuration, un assistant a été implémenté afin de détecter les quelques mots clés se cachant dans la définition naturelle du problème et de les organiser selon le formalisme que la machine utilise pour résoudre automatiquement le problème. Le rôle de ce sous-assistant est avant tout d'aider le concepteur à formaliser le problème. L'automatisation complète n'était pas notre but.

4.3 L'assistant *Param-Def*

Lorsqu'un problème de conception devient de taille importante, sa définition et sa résolution deviennent pratiquement impossible pour un humain. En raison de limitations aussi bien mnésiques, que perceptives ou attentionnelles, il est difficile pour un humain de considérer un problème dans son ensemble. Il est également ardu de se concentrer au moment opportun sur la partie appropriée du problème, celle qui va générer des idées, des compromis à faire ou des conflits à résoudre qui soient synonymes d'innovation.

Les techniques de l'intelligence artificielle pour la résolution de problèmes sont bien développées. En outre, la puissance des machines a considérablement augmenté. Les ordinateurs peuvent être ainsi de bons assistants de calcul pour des humains impliqués dans une tâche de résolution de problèmes. Les techniques de l'IHM (Interaction Homme Machine), de visualisation, permettent également aux machines d'être de bons assistants perceptifs et attentionnels. Comme déjà mentionné, notre objectif est de supporter la définition du problème en complétant les facultés du concepteur et, en particulier, en assistant sa mémoire, son attention, sa calculation et cela grâce aux facultés complémentaires de la machine. Afin de tirer bénéfice de la puissance de calcul de la machine, il est important de définir le problème de conception dans un formalisme mathématique de sorte qu'il puisse être résolu automatiquement par la machine et de sorte que le concepteur puisse bénéficier des facultés de l'intelligence artificielle.

La définition du problème suit la structure d'un problème de satisfaction de contraintes (CSP, pour plus de détails sur ce formalisme, voir [Kumar, 1992]):

- paramètre: un texte comme par exemple le prix, la taille, la couleur, etc.

- domaine: chaque paramètre a une influence sur un domaine. Il peut être soit numérique (discret, par exemple la taille {1, 2, 3, 4, 6}), soit nominal (par exemple la couleur {rouge, bleu, vert}).

- contrainte: c'est une relation imposée entre les paramètres et les domaines, une restriction (couleur == rouge, $(taille > 2) <-> (prix > 100)$, surface == (largeur * longueur), etc.).

Le traitement des contraintes dans la conception est souvent lié à la satisfaction de contraintes, donc aux techniques de l'intelligence artificielle. Il est bizarrement écarté du champ de l'interaction homme machine. ≪Il suffit d'utiliser un simple moteur de résolution de contraintes≫ diront certains chercheurs afin de ne pas aborder leur traitement. Bien qu'à la base de l'interaction entre l'humain et la machine dans une tâche de conception, les contraintes, que nous diviserons en contraintes et critères, sont rarement des données directement manipulables par l'utilisateur. Nous présentons dans ce chapitre la prise en compte des contraintes à tous les niveaux d'une tâche de conception, leur distinction avec les critères et leur importance dans la conception de l'interaction humain-machine. Nous souhaitons ainsi montrer que l'interaction humain machine doit pouvoir avoir lieu au niveau du raisonnement de l'utilisateur.

4.3.1 Un rôle différent pour les contraintes et les critères

Nous montrons dans cette section la différence que nous faisons entre les contraintes et les critères. Les critères sont utilisés pour l'optimisation, pour comparer des solutions. Tandis que les contraintes sont ≪dures≫ et délimitent l'espace de recherche. Il est laissé au choix de l'utilisateur de décider entre contraintes et critères suivant le traitement qu'il désire leur apporter. [Navinchandra, 1991] utilise une représentation différente de la conception. Le problème de conception dans sa globalité est représenté par des contraintes pondérées. La recherche est alors un processus d'optimisation. La meilleure solution est celle satisfaisant le plus de contraintes ≪lourdes≫. Le formalisme utilisé peut être vu dans ce cas comme une satisfaction partielle de contraintes. Cependant, selon nous, il y a souvent des contraintes dans les problèmes de conception qui ne peuvent pas être relaxées et le concepteur a plus souvent besoin d'optimiser des objectifs que des contraintes. Ainsi nous avons choisi de séparer les deux concepts.

Le distinguo entre contraintes et critères n'étant pas clairement défini dans la littérature, il semble nécessaire de le faire. Le petit Robert donne les définitions suivantes: une contrainte est une règle obligatoire et pénible à appliquer. Alors qu'un critère est ce qui sert de base à un jugement d'appréciation. C'est un caractère, un signe

qui permet de distinguer une chose, une notion; de porter un jugement sur un objet. D'une façon plus sommaire, une contrainte est satisfaite ou non alors qu'un critère peut prendre toute une gamme de valeurs d'appréciation. Par exemple, si l'on considère le prix comme étant une contrainte, l'utilisateur pourra dire qu'il veut un produit ne dépassant pas 40.000 francs. A ce moment là, toutes les solutions dépassant 40.000 francs seront évitées. Par contre, si l'on considère le prix comme étant un critère, aucune solution ne sera interdite pour des raisons pécuniaires. Le concepteur aura la possibilité de comparer les solutions du produit en prenant en compte leur prix.

4.3.1.1 Intervention des contraintes et critères dans le processus de conception

Une contrainte floue est une règle peu précise. Néanmoins cela reste une contrainte alors qu'un critère permet d'évaluer. Ainsi, dans la conception, les contraintes permettront de définir, de contraindre, un espace de solutions et les critères permettront d'évaluer cet espace. Nous allons voir au long de ce chapitre les différents niveaux auxquels interviennent contraintes et critères dans une tâche de conception. Notons que pour que la notion de critère soit intéressante il faut prendre en compte un raisonnement plus fin que celui des contraintes. Il n'y a pas de solution idéale à un problème, mais un ensemble de bonnes solutions. Certaines sont meilleures selon certains critères et d'autres le sont pour d'autres. C'est en raffinant incrémentalement sa définition des contraintes et des critères, en interagissant avec le système, que l'utilisateur va atteindre sa meilleure solution. Par exemple, voici quelques attributs (critères et contraintes) imaginables dans le cas de la conception d'une cuisine: topographiques, d'environnement (murs, plan, ...), financières (budget), esthétiques, utilitaires, temporelles (temps de construction, de livraison, ...), etc.

Comme nous venons de l'expliquer, chaque attribut peut aussi bien être une contrainte qu'un critère; c'est au choix de l'utilisateur. Cependant, il nous semble que les attributs liés à l'environnement (plan de la cuisine, meubles déjà existants) sont le plus souvent des contraintes; il est difficile de modifier la forme de la pièce ou de ne pas tenir compte des meubles déjà existants. Il se peut néanmoins que des solutions créatives passent par la violation de ces contraintes. Supposons que le client veuille une cuisine spacieuse alors que sa pièce est petite, il faudra par exemple casser un mur pour satisfaire son critère. Il est donc important de laisser l'utilisateur libre de classer lui-même les attributs du noyau fonctionnel en contraintes et critères, sachant que l'interaction avec la machine l'amènera sûrement à réviser ses choix ou à en définir de nouveaux.

Les notions de contraintes et de critères interviennent à tous les niveaux de la conception. Elles peuvent être plus ou moins cachées à l'utilisateur selon le contrôle que l'on souhaite lui allouer. Les contraintes sont généralement plus importantes dans les phases de définition de la conception alors que les critères interviennent plutôt dans les phases d'évaluation. Cependant, définition et évaluation doivent être dans une même boucle que l'utilisateur va parcourir de nombreuses fois. Ainsi définition et évaluation ne représenteront plus qu'un seul incrément dans un processus de conception.

4.3.1.2 Les contraintes: résoudre ou s'échapper d'un espace sur-contraint

Comme nous l'avons déjà dit, la conception créative provient de deux types de processus cognitifs: un processus consistant à faire des compromis (TradeOff) et un processus basé sur la découverte (Break-through). Chacun de ces processus provient, à notre avis, d'une autre distinction entre deux types extrêmes de problème: des problèmes soit sous-contraints, nécessitant de faire des compromis, soit sur-contraints, où une mutation est utile pour s'échapper. Dans les problèmes sur-contraints, il n'y a aucune solution. Les concepteurs doivent détecter les conflits et les résoudre. Résoudre des problèmes sur-contraints exige deux facultés: de la puissance de calcul pour découvrir les violations les plus importantes et de la créativité pour agrandir l'espace des solutions et pour éviter ainsi les conflits. La limitation de la mémoire humaine est dans le premier cas un problème. Traiter environ cent contraintes est difficile et la découverte de l'évasion exige de l'ingéniosité; le créateur doit s'extraire de l'espace de solutions infructueux.

4.3.1.3 Les critères: trouver la ≪bonne≫ solution parmi ... une infinité?

Dans les problèmes sous-contraints, de nombreuses solutions sont possibles. Il est important que ces solutions puissent être comparées. Souvent, les solutions sont en compétition; elles font partie d'un ensemble appelé Pareto optimal. Des solutions sont bonnes suivant certains aspects mais pas pour tous et ne peuvent donc pas dominer les autres. De la créativité est nécessaire pour découvrir de nouveaux critères et ainsi réévaluer l'ensemble des solutions Pareto optimales. ≪Voir les choses sous un autre angle≫, c'est ce que l'on fait souvent dans la vie de tous les jours pour prendre des décisions. Il faut faire des compromis afin de trouver la solution optimale. Cependant, lorsque le nombre de critères est grand, les choix sont difficiles. Le système d'aide à l'évaluation des solutions doit permettre de visualiser les solutions selon différents critères et ainsi de trouver des compromis dans le cas où ceux-ci sont antagonistes. La machine permet ainsi d'augmenter les facultés perceptuelles de l'utilisateur avec différentes visualisations. Nous verrons plus loin

l'aide qu'apporte l'assistant TradeOff et sa coopération avec l'assistant *Param-Def.*

4.3.1.4 Synthèse

La prise en compte des contraintes et des critères dans un système d'aide à la conception ne semble pas devoir se résumer à un simple système de résolution de contraintes. Le processus de conception consiste justement à définir contraintes et critères, ce qui ne se fait pas d'un seul coup et qui n'a rien d'une tâche automatique. Les critères tout spécialement sont les seuls outils d'évaluation du concepteur. Cependant, même si les objectifs sont clairs a priori, il lui faut consacrer beaucoup d'efforts pour trouver les critères leur correspondant. S'il semble naturel à la communauté IHM de soutenir les phases de définition, d'exploration et de décision liées à la conception, l'étude du traitement des contraintes et des critères devrait être aussi l'une de leur préoccupation afin de trouver les objets d'interaction communs entre l'homme et la machine.

D'après nos observations, les objectifs dynamiques et mobiles ou les espaces mobiles de la recherche, sont la caractéristique principale de la conception créative. Cette hypothèse vient de la distinction que nous faisons entre un processus de résolution de problèmes et un processus de conception, qui lui implique des qualifications créatives. Dans un processus de résolution de problème, l'espace de recherche ne se déplace pas très souvent. Dans un processus de conception, il change jusqu'à ce que l'espace but soit atteint. Pour cette raison, notre système de conception assistée par ordinateur essaie de suivre cette dynamique avec une architecture ouverte en utilisant l'aide de *Param-Def* comme le squelette autour duquel l'exploration s'articule. Contraintes et critères sont définis à un niveau global et sont partagés par tous les sous-systèmes intelligents, qui correspondent à chacune des tâches de la conception. Un système en tâches hiérarchiques ne poserait pas de problèmes supplémentaires pour la prise en compte des critères et des contraintes. Néanmoins, il contredirait le déroulement ouvert d'une tâche de conception.

4.3.2 Formalisme

Les systèmes existants dans le domaine du CSP utilisent généralement un formalisme binaire, basé sur des matrices. Cependant, les matrices ne permettent pas de définir des formes du type: $\ll x == y + z \gg$, $\ll$si x alors (y et z)$\gg$, etc. Afin de pouvoir définir ces règles, qui sont souvent employées par les concepteurs, nous avons construit un formalisme et des algorithmes capables de traiter des contraintes n-aires. En fait, une projection de contraintes n-aires sur des contraintes binaires créerait des solutions supplémentaires incorrectes et détruirait ainsi la validité de notre ensemble de solutions. En outre, afin d'être plus proche du langage naturel

des concepteurs, nous avons choisi de définir les contraintes sous forme logique. Elles permettent de définir des comparaisons, des implications, des conséquences, des équations, etc., les formes souvent utilisées par les concepteurs. Cependant, définir un langage trop large aurait créé de nombreux problèmes: le coût du calcul, du fait du ralentissement de l'interprétation et la vitesse d'apprentissage auraient été plus longue pour l'utilisateur. Nous avons donc gardé le langage aussi simple que possible. La grammaire ci-dessous décrit le langage sous forme récursive. Par exemple, la première règle exprime ce qu'est une contrainte.

```
< contrainte > ::=
        (< contrainte >) - > (< contrainte >) |
        (< contrainte >) < - > (< contrainte >) |
        (< contrainte >) et (< contrainte >) |
        (< contrainte >) ou (< contrainte >) |
        <expression > < predicat > < expression >  |
        Nil |
        T
< predicat > ::=
        < | > | == | = | < = | > =
< expression > ::=
        (< op. Un.> < expression >)|
        (< expression > < op. > < expression>)|
        < parametre > |
        < valeur >
< op. Un.> ::=
        (cos < expression >)|
        (sin < expression >)|
        (tan < expression >)|
        (arctan < expression >)
< op. > ::=
        + | * | / | -
< valeur > ::=
        nombre entier | valeur du domaine d'un parametre.
```

Une des différences conceptuelles principales entre le CSP et le CLP (programmation à base de contraintes logiques) est que le premier est une collection d'algorithmes et de techniques qui peuvent être mis en application dans n'importe quel langage, tandis que le dernier est une classe de langages, avec des règles de calcul ([Jampel, 1996a]). Les langages à base de contraintes sont définis en respectant les règles pour construire les formules avec le but d'exprimer des contraintes. Alors que les CSP se focalisent sur l'implémentation d'un algorithme pour résoudre des contraintes sur ce langage, c'est-à-dire pour décider si un ensemble de contraintes est consistant, ou pour ramener des contraintes à une forme qui peut être résolue. Le formalisme de notre système est quelque part dans l'intervalle. Nous avons conservé les avantages du CSP pour ses nombreux algorithmes que nous avons voulu

disponibles pour l'utilisateur, auxquels nous avons ajouté des visualisations interactives de leurs cheminements. Et nous avons, par ailleurs, essayé de construire un formalisme logique, aussi simple que possible, pour définir les contraintes, comme dans les CLP [Abderrahamane and al, 1994], afin de faciliter l'expression de l'utilisateur.

4.3.3 Expression et visualisation des contraintes: un problème d'IHM

L'utilisateur doit pouvoir accéder à la définition des contraintes à chaque fois qu'il le désire. Ce qui signifie que la définition des contraintes n'est pas statique; elle n'est pas présente juste une fois après l'analyse du concepteur. La définition est dynamique, dans le sens que le concepteur va modifier son problème à chaque fois qu'il ne sera pas satisfait par les résultats, soit parce qu'il n'y a aucune solution , soit parce qu'il y en a trop soit enfin parce qu'elles ne correspondent pas à ce qu'il attendait. Néanmoins, c'est bien l'utilisateur qui fait intervenir la dynamique dans le réseau de contraintes. Il n'y a pas d'algorithme génétique, de réseau de neurones ou de système de raisonnement à base de cas permettant de faire muter le problème vers la bonne solution. Nous croyons au pouvoir de la visualisation et de l'interaction qui peut stimuler le concepteur et augmenter ses capacités dans la résolution de problèmes. De la même façon, nous verrons par la suite que l'utilisateur doit pouvoir accéder à la définition des critères à chaque fois qu'il le désire. La gestion des critères est néanmoins différente de celle des contraintes; elle nécessite l'existence de solutions et se base sur l'évaluation. Notons, de plus, qu'il doit y avoir différents types d'interfaces entre l'utilisateur et le système de résolution de contraintes. Fournir à l'utilisateur une représentation sous forme logique de règles n'est pas suffisant car non intuitif. La visualisation est particulièrement importante dans la conception. Il est donc important de soutenir l'utilisateur le plus tôt possible avec des visualisations plus ou moins réalistes du produit final. Cependant, la représentation graphique n'est pas toujours adaptée pour définir certaines contraintes. Définir par exemple que la racine carré d'un paramètre doit être inferieure à la somme de deux autres paramètres peut être difficile à définir graphiquement. Il est donc nécessaire de fournir à l'utilisateur un éditeur mixte, aussi bien graphique que textuel pour la définition de règles plus abstraites. Nous ne discuterons pas ici des modalités qui peuvent être utilisées par l'utilisateur pour communiquer ses contraintes et ses critères à la machine. Remarquons, néanmoins, que certaines modalités peuvent s'avérer meilleures que d'autres en fonction de la précision de la définition. Par exemple, il est possible d'imaginer une phase d'initialisation du système où l'utilisateur pourra esquisser à l'aide d'un crayon le produit qu'il désire. Cette phase graphique correspond à l'esquisse de la scène pour la mise en place des éléments sans qu'il soit nécessaire de les spécifier complètement. Il est aussi possible

d'utiliser la voix, le geste ou la réalité virtuelle.

4.3.4 Assistants visuels génériques

En plus des éditeurs spécifiques aux domaines abordés et afin de définir une conception, nous avons choisi d'utiliser une représentation générique mixte qui combine règles logiques et contraintes définies graphiquement avec des matrices, par exemple, pour les cas des contraintes binaires.

4.3.4.1 Définir

Règles logiques
Un éditeur permet à l'utilisateur de définir paramètres, domaines et contraintes. Chaque entité peut-être déplacée (important lors de la recherche), modifiée, ou effacée en utilisant des boutons (à l'extrême gauche de la fenêtre de l'assistant *Param-Def* (figure 4.1). Les règles permettent au concepteur de définir ses connaissances et également d'obtenir des données implicites contenues dans la conception.

Figure 4.1: Définition des paramètres, domaines et contraintes logiques.

Aide
Puisque l'écriture de règles peut être une tâche difficile pour un créateur, l'assistant de *Param-Def* supporte ce processus en fournissant à l'utilisateur un menu contenant tous les types possibles de règle et de leur traduction intuitive.

Matrices
Pour ne pas limiter l'utilisateur à un formalisme, le *Param-Def* fournit également des matrices graphiques facilement définissable à l'aide de la souris. De cette façon,

l'utilisateur peut permettre des couples de valeur en cliquant dans la case correspondante de la matrice. Les matrices sont définissables dans la fenêtre de *Param-Def* (figure 4.2).

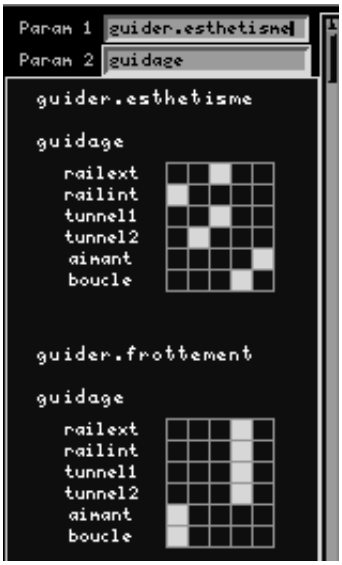

Figure 4.2: Définition graphique d'une matrice

4.3.4.2 Visualiser

Graphe

L'assistant *Param-Def* fournit également une visualisation du graphique créé par des paramètres et des contraintes. Ce graphique est automatiquement mis à jour pendant que les règles sont écrites. C'est un bon ≪feedback≫ visuel pour l'utilisateur parce qu'il donne une vue d'ensemble du problème comme réseau de contrainte. De cette façon, le *Param-Def* sert d'aide attentionnelle parce qu'il aide l'utilisateur à synthétiser ses idées. C'est aussi une bonne façon pour l'utilisateur de visualiser la forme de son réseau de contraintes. Les cycles, les hiérarchies ou les régions isolées apparaissent alors. Les noeuds (les cercles sombres, bleus en fait) représentent les paramètres du CSP tandis que les carrés représentent les contraintes (jaunes si elles sont définies avec des règles, vertes à l'aide des matrices). L'utilisateur peut ≪zoomer≫ sur une partie du problème pour focaliser son attention. Il peut aussi sélectionner à l'aide de la souris un arc afin de connaître la contrainte associée dans l'editeur et de même il peut cliquer sur un noeud et visualiser le paramètre associé.

Transfert règles, représentation graphique

Les matrices sont automatiquement transformées en règles logiques et vice versa. L'utilisateur peut sélectionner, à l'aide de la souris, un ensemble de règles logiques

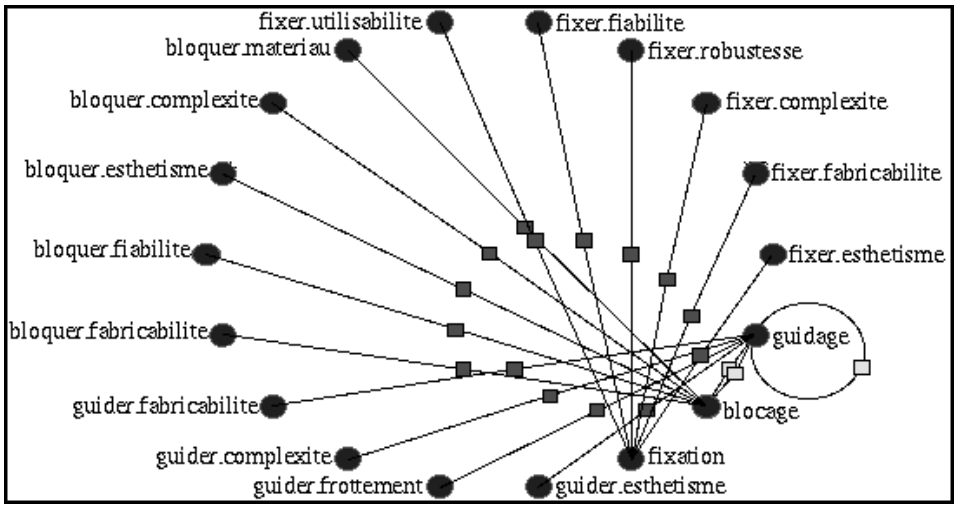

Figure 4.3: Représentation du problème sous forme graphique.

et obtenir sa traduction graphique (matricielle pour un couple de paramètres) automatiquement. Par exemple, la figure 4.4 représente la traduction d'un ensemble de contraintes ne possédant pas plus de deux paramètres distincts. La représentation est donc une matrice. Les carrés sombres, rouges en réalité, représentent les couples interdits par l'ensemble de contraintes sélectionné. La figure 4.5 est un exemple de traduction d'un ensemble de contraintes possédant plus de deux paramètres distincts. La représentation est circulaire et la profondeur du cercle représente le nombre de paramètres distincts dans l'ensemble de contraintes sélectionné. Le premier cercle, à l'intérieur, est divisé en autant de parties que d'éléments dans le domaine du premier paramètre concerné. Chacune de ces parties est elle-même divisée dans le second cercle en autant de parties que contient le domaine du deuxième paramètre, etc.

Figure 4.4: Représentation graphique d'un ensemble de contraintes possédant deux paramètres. Les carrés rouges (les plus sombres) sont les couples interdits.

4.4 Exemple de méta *Param-Def*

Puisque définir un CSP peut être artificiel pour un concepteur, nous avons établi un exemple d'interface plus proche du modèle de l'utilisateur afin de définir les

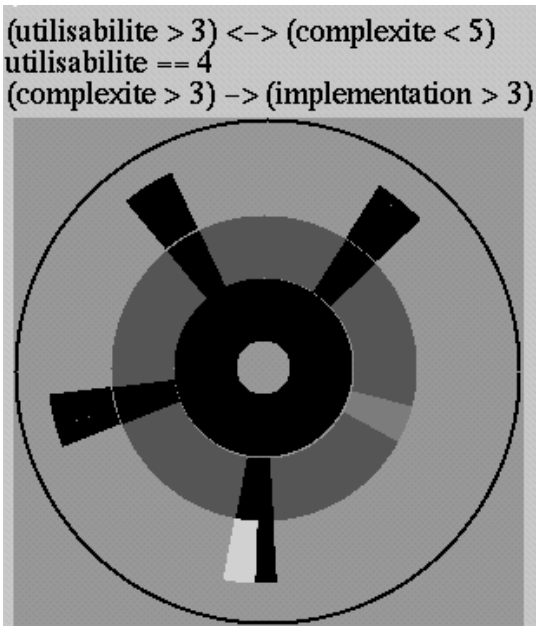

Figure 4.5: Représentation graphique d'un ensemble de contraintes possédant n paramètres ($n > 2$)

structures parallèles de robot. Le formalisme logique pour définir des contraintes est remplacé par des glisseurs (≪sliders≫) et de simples boutons. [Shneiderman, 1990] a utilisé une telle métaphore pour explorer une base de données de films. De la même manière, nous voulons réaliser des interfaces transparentes afin d'améliorer l'interaction humain ordinateur. L'ambition principale de notre travail, encore une fois, est d'établir une synergie entre l'humain et la machine dans une tâche de raisonnement: la résolution des problèmes. Les glisseurs sont directement reliés au *Param-Def* comme il est possible de le voir sur la figure 10.1 (page 100). Chaque fois que le concepteur modifie une contrainte à l'aide d'un glisseur, la définition dans *Param-Def* est automatiquement mise à jour. En outre, quand le concepteur ne trouve pas toutes les requêtes souhaitées dans cette méta interface, il peut entrer dans le *Param-Def* et ajouter une contrainte logique ou matricielle ou même un nouveau paramètre. Tous les aides de Comind sont accessibles depuis le Méta-*Param-Def*. Ce méta est juste un module connectant le concepteur du robot au *Param-Def* par une voie plus naturelle.

Chapitre 5

Recherche de solutions

≪Résoudre un problème complexe n'exige pas seulement de l'ingéniosité. Quand le problème est grand, il exige également de la puissance de calcul pour le traitement de l'information≫.

5.1 Introduction

La phrase par laquelle nous commençons ce chapitre illustre l'une de nos hypothèses principales que nous appliquons ici à l'assistant *Solve*. Selon nous, augmenter la taille de l'espace de recherche parcouru, avec une aide de calcul, améliore également les performances et la qualité du processus de résolution de problèmes. ≪Afin de découvrir une belle idée, il faut en explorer de nombreuses≫ [Fischer, 1993].

Nous présentons dans ce chapitre un ensemble d'outils supportant la recherche des solutions pour un problème de satisfaction de contraintes (CSP). Cet ensemble est composé d'aides au calcul et à la réflexion. Ces outils coopèrent avec l'utilisateur afin de résoudre le problème. L'organisation de cet ensemble de sous-assistants et leur interaction visuelle avec l'utilisateur sont particulièrement illustratives de notre approche et nous insisterons ainsi sur ces parties. L'assistant *Solve* est un bon exemple du modèle d'interaction que nous proposons dans cette thèse, basé sur la collaboration intelligente entre l'homme et la machine. En considérant le système de résolution de contraintes comme un assistant de traitement de l'information indépendant des autres fonctionnalités du système, l'utilisateur pourra l'appeler lorsqu'il le souhaite. Il serait dommage de brider la créativité de l'utilisateur avec un système vérifiant que toutes les contraintes sont satisfaites à tous les moments de la conception. Nous avons donc conçu le système pour que le concepteur puisse appeler l'assistant pour vérifier la satisfaction des contraintes quand il le juge nécessaire. L'assistant Solve utilise différents algorithmes de recherche ayant différents degrés d'efficacité et de rapidité. Il fournit aussi différentes visualisations de l'espace de

recherche. En effet, notre système de résolution de contraintes propose une visualisation de l'espace de recherche ainsi que du parcours des différents algorithmes de recherche afin de donner un plus grand contrôle à l'utilisateur. Le concepteur peut ainsi interactivement explorer l'espace de solutions; la machine augmentant ainsi les facultés de traitement de l'information du concepteur. Remarquons que tous nos assistants du processus de conception possèdent aussi bien des sous-assistants de calcul que des sous- assistants d'interfaçage ou de visualisation. L'assistant Solve est un bon exemple d'outil de recherche interactif car l'utilisateur dispose de plusieurs assistants de recherche et de plusieurs assistants de visualisation. Il est libre de faire coopérer ses outils de recherche au travers des visualisations, ou de les faire fonctionner dans un mode concurrent.

5.2 L'architecture de l'assistant *Solve*

La mise en place des assistants a été faite selon leur rôle et leurs interactions avec l'utilisateur. Les assistants de calcul sont implémentés en LISP tandis que les assistants réflexifs sont codés en Tcl/TK. Puisque les assistants de calcul sont consacrés aux tâches de recherche et parce qu'ils peuvent être exécutés en parallèle, le LISP est un langage particulièrement adapté. De la même manière, parce que les assistants réflexifs sont consacrés à la communication avec l'utilisateur et entre les aides et également à la visualisation, Tcl/TK est l'outil parfait pour l'interfaçage et le script.

Le dispositif important de cette architecture est que l'utilisateur possède un contrôle total des assistants. Le concepteur peut visualiser les travaux des assistants de calcul et leurs communications, au travers des assistants réflexifs, les arrêter et focaliser leurs recherches. L'utilisateur explore ainsi l'espace de recherche comme s'il était aux commandes d'un vaisseau spatial, assis dans un cockpit [Spence et al., 1995].

5.3 Des assistants pour traiter l'information

L'interaction d'un utilisateur typique avec notre système consiste à utiliser d'abord la consistance des noeuds et puis la consistance des arcs. L'utilisateur évalue alors le nombre de solutions que le problème possède en utilisant l'algorithme de Knuth. L'utilisation d'un assistant spécifique est suggérée par invitation (≪affordance≫). A chaque étape de l'interaction, les meilleurs assistants sont suggérés à l'utilisateur. Cependant, l'utilisateur est toujours libre de décider de sa voie d'action.

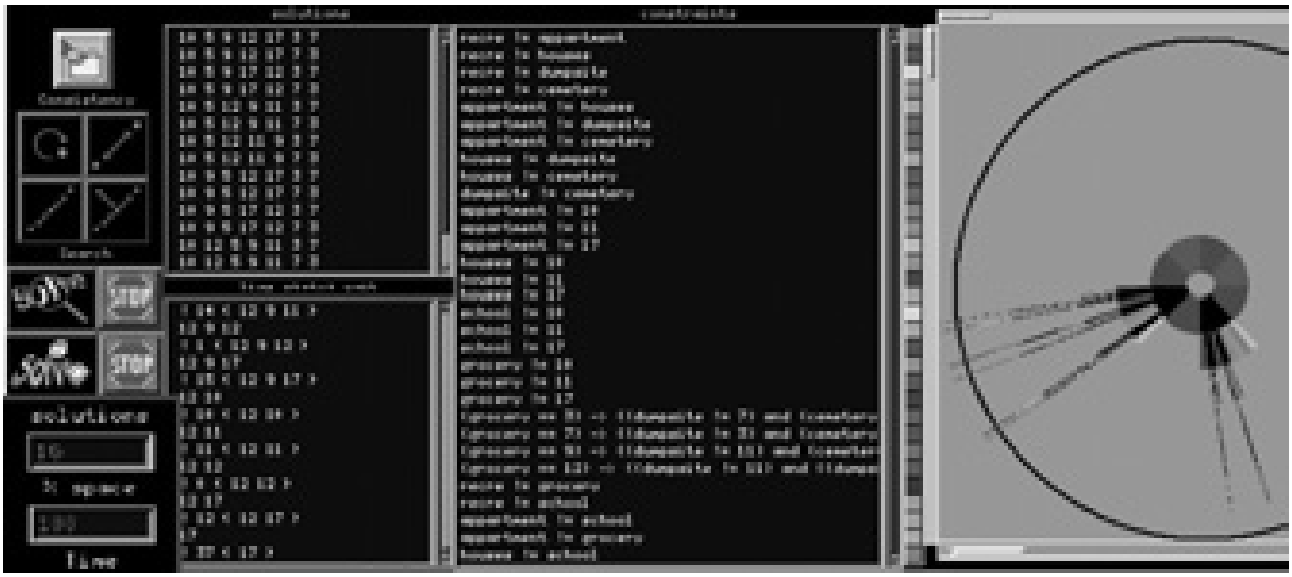

Figure 5.1: L'assistant *Solve* de Comind. En haut à gauche, l'utilisateur peut tester la consistance du problème. En dessous, il peut démarrer deux types de recherche. Le nombre de solutions déjà trouvé, ainsi que le pourcentage de l'espace parcouru et le temps restant, sont finalement indiqués.

5.3.1 Définitions

Une **valuation** est l'assignation d'une valeur à chaque variable du problème; la valeur appartenant au domaine de la variable. Une valuation **satisfait** une contrainte si la contrainte est valide pour cette valuation. Une **solution** à un problème est une valuation qui satisfait toutes les contraintes du problème.

5.3.2 Recherche par retour arrière

Le backtracking consiste en l'instantiation séquentielle des variables suivant leur ordre d'apparition dans leurs domaines. Lorsque toutes les variables de tous les paramètres concernés ont pris une valeur, la valuation ainsi obetnue est testée. Si cette valuation partielle est satisfaite, le processus d'instantiation et de test continue. Si la valuation partielle est fausse, le processus échoue au niveau du dernier paramètre instantié et qui a des valeurs non essayées dans son domaine; ce paramètre est alors ré-instantié à sa prochaine valeur. Nous avons adapté ce backtracking standard aux contraintes n-aires afin de pouvoir traiter le langage défini dans *Param-Def* (voir section 4.3.2, page 34).

5.3.3 L'aide à la recherche aléatoire

Le rôle de cette aide est d'explorer aléatoirement plusieurs endroits consistants de l'espace de recherche, qui peuvent fournir des solutions, de sorte que l'utilisateur puisse avoir une idée de la forme de l'espace. Ainsi, le concepteur pourra approfondir en démarrant une recherche plus précise sur une partie spécifique de l'espace, à l'aide du ≪backtracking≫ par exemple. Cette recherche est rapide et fournit la morphologie de l'espace de recherche. Cette recherche aléatoire est basée sur l'algorithme de Knuth que nous présentons en détail plus loin (voir section 5.3.5).

5.3.4 Contrôle de consistance

Pour plusieurs raisons, le retour arrière (≪backtracking≫) peut être inefficace. Il y a trois situations qui causent un comportement de destruction pathologique (≪trashing≫ [Jampel, 1996a]):

1. si le domaine Di, de la variable vi, contient une valeur qui ne satisfait pas Pi(x) alors cela sera la cause d'une panne répétée qui pourrait être éliminée en supprimant simplement une fois pour toutes les éléments du domaine qui ne satisfont pas le prédicat unaire correspondant.

2. supposons que les variables soient instantiées dans l'ordre v1, v2..., vn et que vi=a, Pij(a, vj) (où $j > i$) ne soit satisfait par aucune valeur de vj. Alors, le ≪backtracking≫ essayera toutes les valeurs vj, échouera et essaiera toutes les valeurs que peut prendre vj-1 (pour chacune des valeurs de vj) et ainsi de suite jusqu'à ce que toutes les combinaisons des valeurs de vi+1, vi+2..., vj aient été essayées avant finalement de découvrir que a n'est pas une valeur possible pour vi. Ce qui est pire encore est que ce processus de panne pourra être répété pour tous les autres ensembles de valeurs de v1, v2... vi-1 avec vi=a.

3. comme dans le cas précédent, un ≪trashing≫ aura lieu lorsque vi=a, vj=b, Pi(a) et Pij(a, b) sont simultanément satisfaits et qu'ils doivent être présents dans toutes les solutions.

Les termes appropriés pour dénommer chacun des états qui mènent aux trois phénomènes décrits ci-dessus sont respectivement, la consistance des noeud, la consistance des arcs et la consistance des chemins. Afin d'améliorer l'efficacité des assistants de calcul, le contrôle de la consistance est donc généralement le premier travail suggéré. Son rôle est de contrôler la consistance du réseau des contraintes et de réduire ainsi la taille de l'espace de recherche.

5.3.5 Evaluation de la taille du problème à l'aide de Knuth

Le rôle de cet algorithme est d'évaluer l'efficacité des programmes de recherche par retour arrière (≪backtracking≫). L'algorithme de Knuth utilise une approche dite de Monte Carlo, basée sur une exploration aléatoire de l'arbre de recherche [Knuth, 1975]. Pour chaque solution partielle, une continuation valide est aléatoirement choisie. Quand l'algorithme atteint une feuille de l'arbre[1], le nombre estimé de solutions est retourné, en fonction du nombre d'embranchements valides écartés. L'algorithme parcourt uniquement un noeud à chaque niveau de l'arbre de recherche, ce qui le rend très rapide. Si le problème est sous-contraint, le concepteur peut définir davantage de contraintes afin de restreindre l'espace de recherche ou utiliser l'assistant TradeOff, qui est une aide à la comparaison des solutions et à la recherche de l'optimale. Si le problème est sur-contraint, l'utilisateur peut utiliser l'assistant ConflictElicit qui permet de mettre à la lumière les conflits inhérents au problème, s'il y en a, et qui diagnostique les contraintes les plus intéressantes à relaxer ou à repenser. Dans le cas où le problème n'est ni sur, ni sous-contraint, l'assistant *Solve*, que nous présentons dans ce chapitre, possède les outils adaptés à la résolution interactive du problème.

5.3.6 Arrêt

Tous les assistants de calcul sont contrôlables et peuvent être arrêtés. La visualisation est particulièrement importante pour le contrôle de ces algorithmes parce qu'elle indique à l'utilisateur ce que l'algorithme est en train de faire.

5.3.7 Un espace de travail universel pour la recherche

Les assistants de calcul actuellement mis en application représentent un ensemble presque minimal d'algorithmes pour démontrer l'assistant *Solve*. Un projet, actuellement en cours, utilise l'architecture de Comind et tout particulièrement l'espace de travail et de visualisation de l'assistant *Solve*. Il utilise une bibliothèque contenant tous les algorithmes existants de CSP (résolution de problèmes par satisfaction de contraintes) codés en Java (JCL). L'architecture de Comind a particulièrement intéressé les chercheurs pour sa zone de travail qui permet la coopération de l'utilisateur avec plusieurs assistants de calcul grâce à la visualisation. En outre, l'architecture permet d'installer facilement de nouveaux algorithmes en tant que nouveaux assistants. Ceci est particulièrement important pour l'extension et l'adaptation du système. Par exemple, nous prévoyons à l'avenir que l'utilisateur

[1]Une feuille d'un arbre de recherche est un noeud qui ne possède aucun fils; c'est une ramification terminale.

pourra rechercher les algorithmes dont il a besoin à travers le réseau internet, en utilisant l'assistant Solve comme un environnement de travail pour la résolution de problèmes profitant ainsi de toutes les interfaces et visualisations existantes.

5.4 Les assistants réflexifs

5.4.1 Visualisation du contrôle de consistance

Il est possible de voir dans le Param-Def quelles sont les valeurs des domaines qui ont été supprimées. Lorsque le ≪backtracking≫ est démarré ces valeurs ne sont pas testées. Il est donc possible de voir quelles valeurs ont été enlevées. Si la consistance élimine toutes les valeurs d'un domaine, un simple message indique qu'il n'y a aucune solution au problème.

5.4.2 Visualisation de l'estimation

La visualisation de l'algorithme de Knuth est une simple jauge indiquant le niveau du problème par rapport aux trois classes présentées précedement: sur-contraint, normal, sous-contraint. Il faut noter que dans le cas sur-contraint, il est parfois préférable de faire une recherche ou de tester la consistance afin d'être tout à fait sûr que le problème ne possède aucune solution. L'algorithme de Knuth est probabilistique et permet de faire surtout la différence entre un problème possédant un nombre normal de solutions et un problème sous-contraint, qui en possède trop.

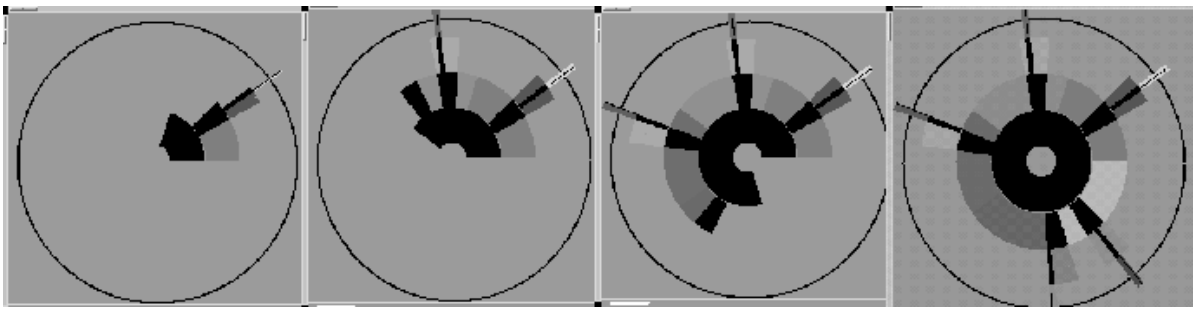

Figure 5.2: Visualisation d'un backtracking. La dynamique de la recherche dans l'assistant *Solve*. Lorsque le cercle extérieur est coupé par une ligne noire, c'est qu'une solution a été trouvée; tous les paramètres ont une assignation correcte.

5.4.3 Visualisation interactive pour contrôler les recherches

Nous avons choisi d'avoir une visualisation commune pour tous les algorithmes de recherche afin de pouvoir utiliser plusieurs algorithmes à la fois. La visualisation est du même type pour tous les aides de calcul. C'est juste une entité qui permet

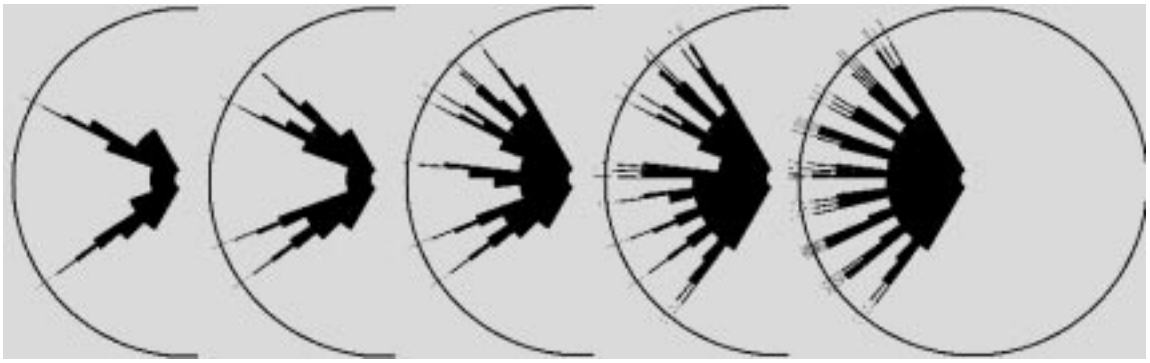

Figure 5.3: Visualisation d'une recherche basée sur l'algorithme de Knuth. La recherche se fait aléatoirement dans l'espace.

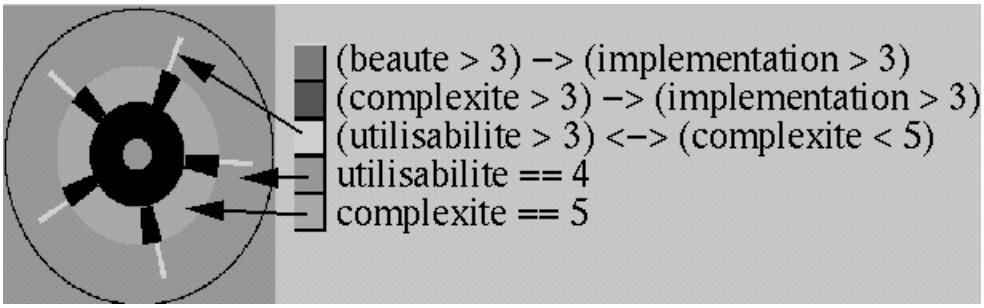

Figure 5.4: L'interactivité de la recherche dans l'assistant *Solve*. Quand l'algorithme n'atteint pas une solution, les contraintes responsables correspondantes sont représentées par une couleur. Lorsque l'algorithme est stoppé en profondeur, la contrainte responsable de cet arrêt est indiquée par sa couleur.

de faire communiquer tous les algorithmes. La visualisation du cheminement des algorithmes se fait en temps réel. Elle est dynamique et peut être stoppée (figure 5.2). Ainsi, par exemple, un type de recherche peut être interrompu afin d'utiliser un algorithme de recherche plus adapté sur une partie plus restreinte de l'espace. Elle permet de visualiser aussi bien le cheminement d'un ≪backtracking≫ intelligent (figure 5.2) que celui d'un algorithme aléatoire (figure 5.2). L'idée principale est de rendre l'utilisateur capable d'utiliser plusieurs algorithmes et de les faire coopérer à travers différentes visualisations. Cela permet aussi au concepteur de savoir pourquoi une zone de l'espace de recherche est interdite, quelles contraintes en sont la cause (figure 5.4). Nous verrons que cette caractéristique de l'assistant *Solve* est le fil conducteur liant l'assistant *Solve* avec l'assistant Conflict. L'utilisateur peut aussi sélectionner à la souris une partie de l'espace sur laquelle il a l'intuition qu'une recherche s'impose. Ainsi le concepteur peut choisir à l'aveuglette un espace de recherche qui lui semble intéressant, faculté essentielle à la base de nombreuses découvertes. L'idée est principalement de rendre l'utilisateur capable de faire appel

à plusieurs types d'algorithmes à la fois en les faisant coopérer par la visualisation interactive. La figure 5.5 montre le backtracking d'un même problème. La visualisation est différente car le concepteur a changé l'ordre des paramètres. Pour cette raison, la synchronisation entre l'assistant *Solve* et l'assistant *Param-Def* est particulièrement importante. Le concepteur doit être à tout moment apte à modifier la définition du problème afin d'intervenir sur la recherche.

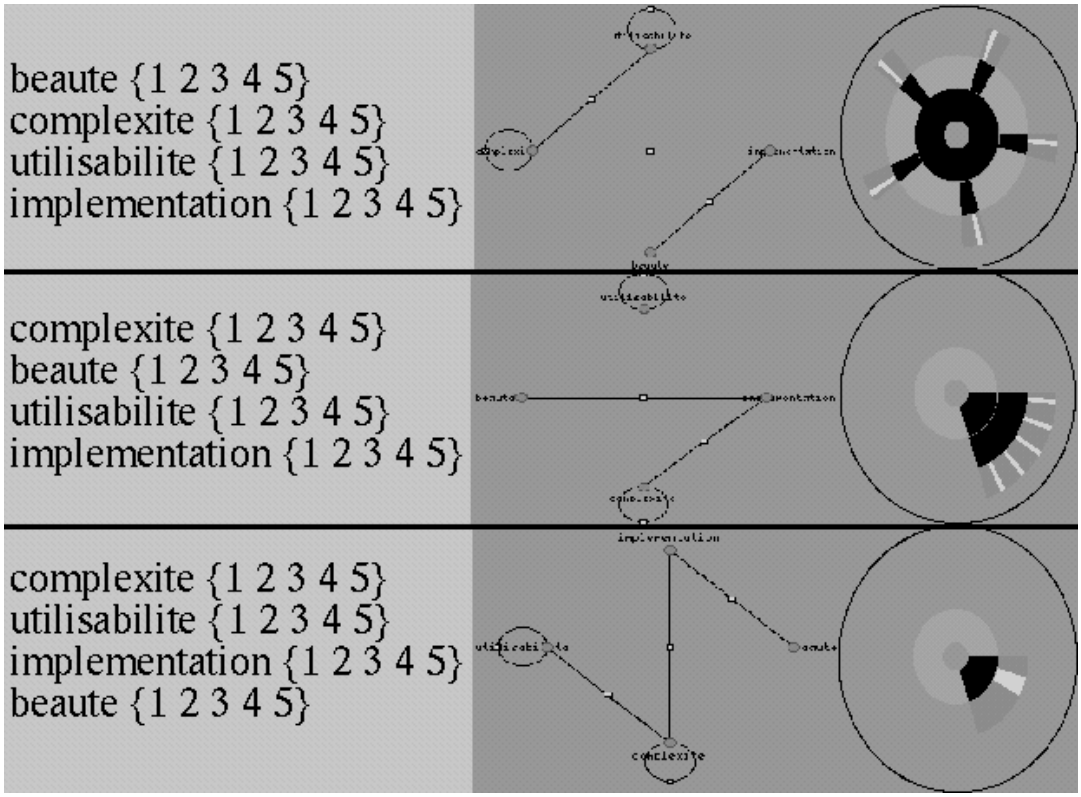

Figure 5.5: Une visualisation de recherches en cours du même problème qui montre l'importance de l'interactivité et de l'ordre des paramètres. Lorsque l'ordre des paramètres change, le backtracking peut être plus ou moins rapide. De bas en haut, sur cette visualisation, l'ordre est de plus en plus optimal et la recherche plus focalisée.

5.5 Un niveau différent de collaboration

Nous avons montré comment, dans l'assistant *Solve*, l'utilisateur coopère avec les sous-assistants. Notons que l'assistant *Solve* coopère de façon différente avec l'utilisateur lorsqu'il est utilisé avec l'assistant *Param-Def*, TradeOff, ou Conflict. Nous verrons dans les scénarios d'interaction à la fin de cette thèse comment se déroule la coopération entre assistants.

Chapitre 6

Rechercher la solution optimale

6.1 Introduction

Lorsqu'un problème de conception possède de nombreuses solutions, il est important que ces solutions puissent être évaluées suivant un ensemble de critères et que les meilleures solutions s'imposent ([Radford and Gero, 1988] et [Navinchandra, 1991]). Souvent, des solutions s'avèrent concurrentes. Aucune d'elles ne domine une autre. Elles sont Pareto optimales ([Pareto, 1896]). C'est-à-dire que certaines solutions sont bonnes sous certains aspects, mais pas sous tous et ne peuvent donc pas dominer les autres. De la créativité est alors souvent nécessaire afin de découvrir de nouveaux critères et faire émerger une solution de l'ensemble Pareto-optimal ou pour réévaluer les solutions sous un aspect différent. C'est ce que nous faisons naturellement dans la vie lorsque nous ≪voyons les choses sous une nouvelle lumière, ou sous un angle différent.≫.

Beaucoup de travaux ont été effectués, dans le domaine de l'optimisation, afin d'établir des programmes capables de résoudre ces problèmes sous-contraints. Cependant, ces programmes n'interagissent que peu avec l'utilisateur et le manque d'intuition des machines ne leur permet pas de produire des compromis[1]. Nous avons déjà expliqué que nous essayons de définir un espace interactif où l'homme et la machine peuvent collaborer efficacement. Dans les problèmes sous-contraints, les concepteurs doivent pouvoir balancer les solutions afin de faire les meilleurs compromis pour trouver la solution optimale. Lorsque le nombre de critères est grand, les choix sont difficiles. C'est à ce point que la machine peut intervenir pour aider l'humain à explorer plus facilement, rapidement et précisement l'espace de solutions en lui fournissant des outils qui améliorent sa perception.

[1]COMPROMIS: n. m., 1. Dr. Convention par laquelle les parties, dans un litige, recourent à l'arbitrage d'un tiers. 2. Arrangement dans lequel on se fait des concessions mutuelles. 3. (Fin XIXe). Fig. état, situation intermédiaire, à moyen terme.

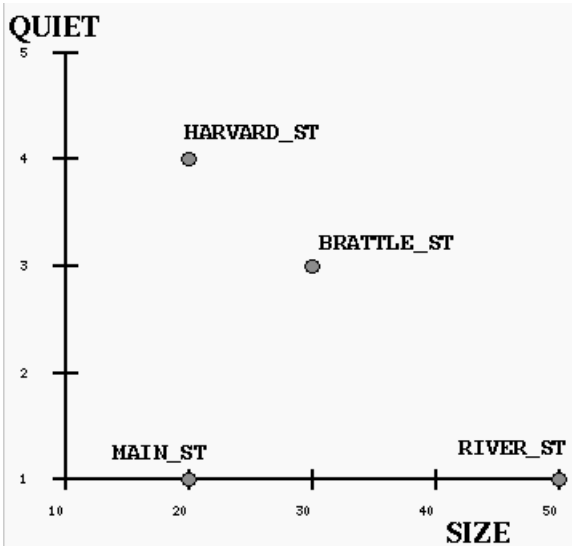

Figure 6.1: Aucun appartement ne domine suivant les deux critères taille et insonorisation.

6.1.1 Un exemple de compromis

Le scénario suivant, inspiré de [Navinchandra, 1991], illustre mieux ce type de problèmes. Par souci de simplicité, nous avons choisi un scénario qui n'est pas une conception à proprement parlé. Cependant, en général, nos techniques sont principalement appliquées au domaine de la conception.

En arrivant dans une nouvelle université à NewYork, un étudiant se trouve face à quatre choix d'appartements. Il possède principalement deux critères, le niveau de bruit et la taille de l'appartement. Il trace les solutions selon ces critères dans le graphique représenté sur la figure 6.1. Trois solutions s'avèrent Pareto optimales mais elles sont équivalentes pour les critères donnés. Dans de telles situations, l'utilisateur manque souvent de motivation pour choisir n'importe quelle solution. Davantage d'explorations avec l'agent immobilier lui ont apporté une cinquième solution, un appartement sur la rue de Brookline qui est actuellement dominé par les autres solutions. Cependant, en même temps, il a découvert des critères d'un tiers pour évaluer ses solutions, la distance entre l'appartement et les transports publics. Le nouvel appartement est très près du métro et donc il est poussé sur la surface Pareto. Maintenant l'utilisateur est prêt à étudier les différences entre ces solutions. C'est-à-dire, étant donné ces cinq possibilités, il doit décider quels critères sont les plus importants pour lui. Puisque la dernière solution a une optimalité bien plus forte que les autres en ce qui concerne le critère de transport, la plus importante pour l'utilisateur, elle a été choisie (voir figure 6.2). L'appartement sur Brookline domine pour les trois critères à la fois. Cependant, cette solution n'est pas optimale pour

chacun des critères pris indépendamment. L'utilisateur a du faire un compromis.

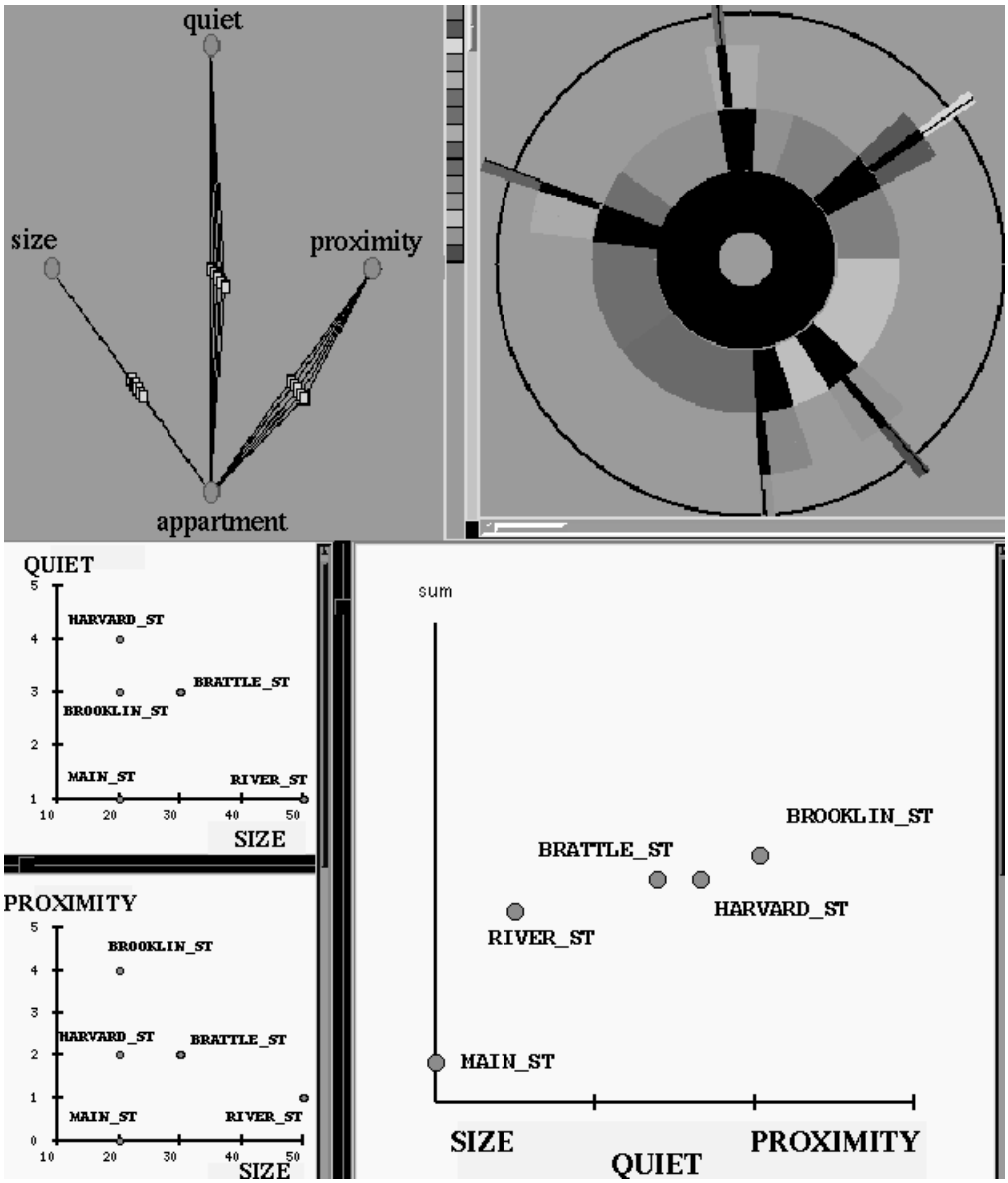

Figure 6.2: L'appartement sur Brookline domine lorsque l'évaluation porte sur les critères de taille, d'insonorisation et de proximité. Cependant, il ne domine ni pour la taille, ni pour l'insonorisation. Un compromis a donc du être fait.

6.1.2 Limitations cognitives

Les humains possèdent des difficultés à visualiser de grands ensembles d'informations, et à en faire le tri. Il y a plusieurs aspects du système cognitif humain qui en sont la cause:

- les images mentales ne correspondent pas simplement à ce que nous voyons. Si ce que nous voyons n'est pas toujours ce que nous avons à l'esprit, notre perception, prise dans le sens global comme un jugement intuitif, peut nous amener vers une conclusion fausse. Par exemple, on a proposé à des sujets d'imaginer qu'ils tiennent un cube par deux des ses coins opposés (le coin inférieur et le coin supérieur sont verticalement alignés). On leur a alors demandé leur avis sur la position des autres coins, ceux qu'ils ne tiennent pas. La majeure partie des sujets pense que les autres coins sont alignés sur le même plan horizontal correspondant à l'équateur du cube. En réalité, les coins du cube forment une couronne et ne sont pas alignés sur le même plan. Pour cette raison, des visualisations différentes d'un même problème peuvent permettre de minimiser les chances de mal interpréter la réalité.

- il est également connu que la mémoire à court terme humaine est limitée. Un humain ne peut mémoriser que 4 à 7 gros morceaux conceptuels (des ≪chunks≫). En outre, il est souvent difficile de visualiser un ensemble de vecteurs en trois dimensions. Dans des dimensions plus élevées, cela devient presque impossible.

Pour ces raisons, les visualisations que peut fournir l'ordinateur peuvent être de bons outils pour améliorer les facultés perceptuelles de l'humain.

6.1.3 Quelques définitions

Tradeoff
Un ≪trade off≫ est un balancement. C'est l'action de faire des compromis.

Slipoff
Un ≪slip off≫ est un glissement, un déplacement. Lorsque l'utilisateur possède l'intuition qu'une solution peut être bonne, il est possible de la rendre optimale en la faisant glisser vers l'espace pareto-optimal en ajoutant un critère par exemple. Cela peut correspondre aussi à la relaxation d'une contrainte ou à la diminution de l'importance d'un critère.

Mutation
Une mutation (Cf section 2.2.4, page 11) est un saut, un changement d'espace ou un changement de direction dans l'espace de conception. Cela peut correspondre à l'ajout d'un paramètre ou en général à la reformulation du problème.

6.1.4 Importance de l'interaction humain-machine

Nous croyons que l'interaction humain machine est une condition essentielle pour bien réussir un tâche d'optimisation multicritère. Les critères supplémentaires (phénomène émergent, voir Tomlin [Tomlin, 1986]) sont suggérés par les utilisateurs pendant l'exploration liée à la conception. Les machines, cependant, stimulent l'humain dans l'apparition des critères en montrant des cas de conception ou en fournissant simplement des données multimédias de produits (voir section 8.2, page 86). Les machines fournissent également des visualisations de l'espace des solutions suivant les critères de sélection. Les solutions peuvent être, par exemple, tracées dans un espace tridimensionnel comme montré dans la figure 6.7. Une surface Pareto peut être construite pour illustrer les solutions qui dominent. Les solutions dominées sont éliminées ou bien, les utilisateurs peuvent essayer d'ajouter des critères supplémentaires de sorte que certaines d'entre elles puissent s'élever sur la surface Pareto (≪slip off≫). En outre, n'importe quelle solution peut être sélectionnée à la souris et ainsi être comparée à d'autres visualisations du même espace de solutions suivant un ensemble différent de critères.

6.2 Références

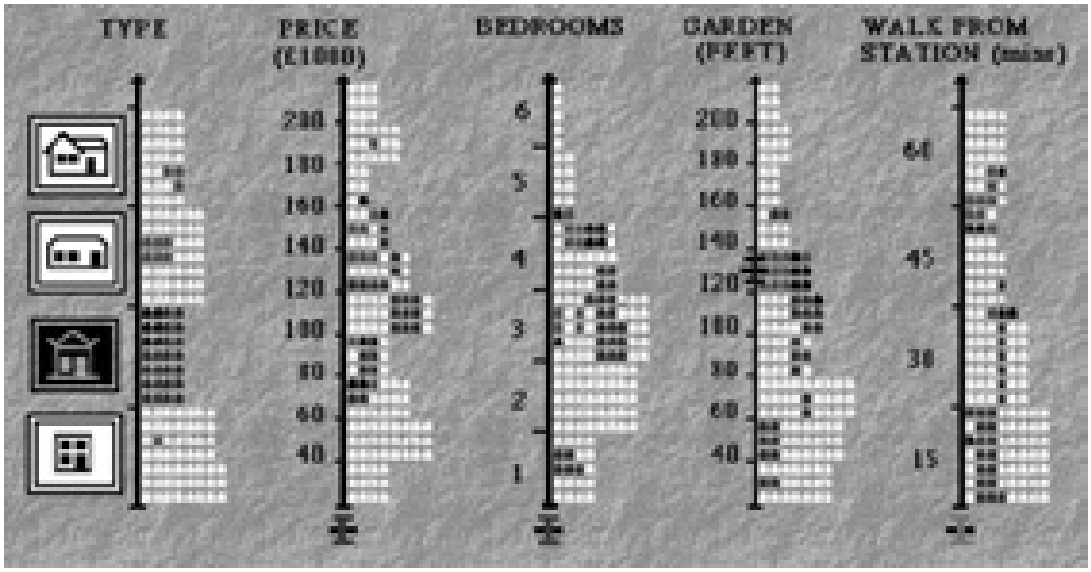

Figure 6.3: L'≪attribut explorer≫ de [Tweedie et al., 1994] permet de visualiser les dépendances entre les paramètres d'un problème. Dans l'exemple, pour un type de maison particulier, la visualisation permet de visualiser les valeurs que peuvent prendre les paramètres: prix, nombre de chambres, taille du jardin et distance par rapport à la gare.

L'≪attribut explorer≫ et l'≪influence explorer≫ sont des applications qui permettent de visualiser les relations entre les paramètres d'un problème (respectivement [Tweedie et al., 1994] et [Tweedie et al., 1996]). L'interactivité est une des caractéristiques les plus importantes de ces systèmes, permettant à l'utilisateur d'explorer l'espace des solutions (figure 6.3). Une des hypothèses principales de ce travail est que ces visualisations interactives permettent d'extérioriser les modèles mathématiques abstraits, implicites par exemple à une conception. Il est montré aussi comment l'abstraction peut aider à informer sur la conception. Ce travail est particulièrement intéressant pour son aspect interactif et visuel car il permet de faire apparaître les dépendances entre paramètres. Cependant, il n'aborde pas le processus de résolution de problèmes dans sa globalité et ne s'intègre pas dans un système possédant d'autres outils (afin de définir, résoudre, relaxer, etc); il ne traite pas non plus de réels problèmes de conception.

6.3 L'assistant TradeOff

6.3.1 L'éditeur de critères de l'assistant TradeOff

L'assistant TradeOff fournit un éditeur pour définir les critères de la conception, comme le Param-Def pour les contraintes. Le calcul de ces critères se fait alors que les solutions sont déjà trouvées. C'est une phase post-recherche. Les solutions vont être évaluées selon ces critères. Ceux-ci sont calculés pour toutes les solutions trouvées pendant le processus de recherche. Plusieurs visualisations sont proposées au concepteur afin d'évaluer les solutions selon les critères définis dans l'éditeur. Le concepteur peut ainsi évaluer la pertinence du choix de ses critères.

6.3.2 Les assistants réflexifs

Le rôle de l'assistant TradeOff est d'aider le concepteur à trouver la solution optimale dans un problème sous-contraint. L'utilisateur définit des critères afin de comparer les solutions. Nous proposons plusieurs visualisations qui permettent à l'utilisateur de voir les solutions sous différentes lumières et de les manipuler. Nous présentons dans cette section les différentes visualisations grâce auxquelles l'utilisateur peut explorer l'espace des solutions.

L'utilisateur peut choisir le type de visualisation qu'il désire avec les critères qu'il souhaite. Les visualisations sont interactives; sélectionner ou déplacer un objet sur une visualisation affecte immédiatement les autres visualisations. Les visualisations sont donc toutes liées entre elles. L'important dans ce type de tâche est d'aider l'utilisateur à jongler facilement avec plusieurs critères afin qu'il trouve la solution

optimale. L'interaction visuelle est particulièrement importante car elle permet, en interagissant avec différentes vues, de trouver la meilleure solution suivant certains critères. De plus, lorsque l'utilisateur a l'intuition qu'une solution peut être bonne, il peut chercher les critères qui la rendent optimale. Le concepteur peut également évaluer ainsi la qualité et l'acuité des critères.

6.3.2.1 La solution dans son contexte

L'assistant TradeOff permet de visualiser une solution dans son contexte le plus figuratif. La figure 6.4 montre quelques exemples de visualisations dans notre système comme par exemple la configuration d'une cuisine, d'un robot parallèle et du plan d'un lotissement. A l'extrême gauche de cette figure, un exemple de mauvaise conception est représenté; c'est la photo d'une théière pour masochiste [Norman, 1988]. Cette photo n'a pas été générée par notre système. Cependant, c'est un exemple qui illustre bien ce qui arrive souvent dans notre système; des cas particulièrement créatifs peuvent être découvert en parcourant l'espace des solutions.

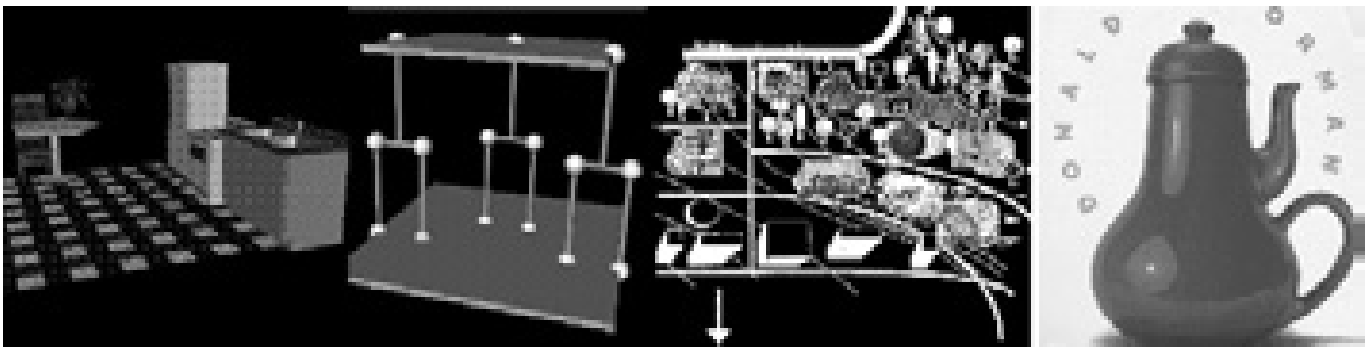

Figure 6.4: Quelques exemples de visualisations de solutions dans leur contexte.

6.3.2.2 Echelle

La visualisation 6.5 représente la valeur relative des critères entre leur valeur minimale potentielle, et leur valeur maximale possible dans l'espace de toutes les solutions. Par exemple, la figure 6.5 indique que cette solution de configuration de cuisine possède le prix le plus bas par rapport aux autres solutions trouvées, un espace assez grand et une distance de la table au frigidaire moyenne.

6.3.2.3 Parato 2D

Cette visualisation (figure 6.6) représente la paréto-optimalité pour un couple de critères. Une solution est considérée pareto-optimale s'il n'est pas possible de trouver une solution meilleure pour les deux mêmes critères à la fois. (a,b) est pareto-optimale si et seulement si $\neg\exists$ de (xi, yi) tel que $xi > a$ et $yi > b$.

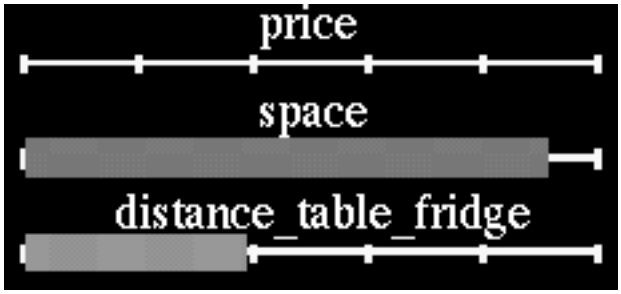

Figure 6.5: Une échelle de mesure. Le prix est ici minimal, l'espace presque maximal et la distance entre la table et le frigidaire est moyenne par rapport aux autres solutions.

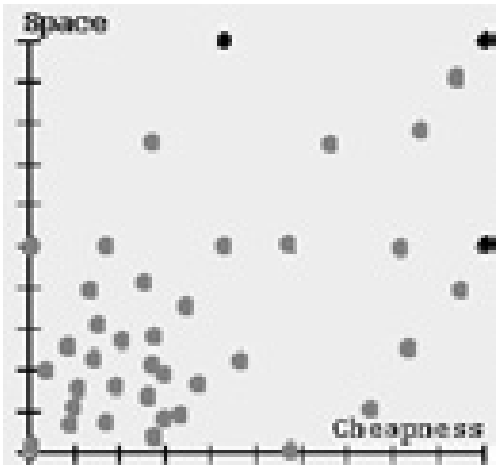

Figure 6.6: Comparaison de deux critères, paréto-optimalité (Pareto 2D). Les solutions Paréto ne sont dominées par aucune autre.

6.3.2.4 Parato 3D

La visualisation 6.7 représente la paréto-optimalité pour trois critères. La pareto-optimalité pour 3 critères se défini ainsi: (a,b,c) est pareto-optimale si et seulement si $\neg\exists$ de (xi, yi, zi) tel que $xi > a$ et $yi > b$ et $zi > c$. Pour cette visualisation, nous avons représenté en trois dimensions les critères. Pour faciliter la lecture à l'utilisateur, plus la distance à l'origine est grande, plus la solution, la sphère dans le repère en trois dimensions, prend une couleur foncée. Ainsi, les solutions optimales sont les solutions les plus proches de l'observateur et aussi les plus foncées. Les pareto-optimales ne sont dominées par aucune autre pour les trois critères.

6.3.2.5 Parato 4D

La visualisation 6.8 représente la paréto-optimalité pour quatre critères. Il y a quatre repères en trois dimensions. Chaque repère est l'une des combinaisons possibles de

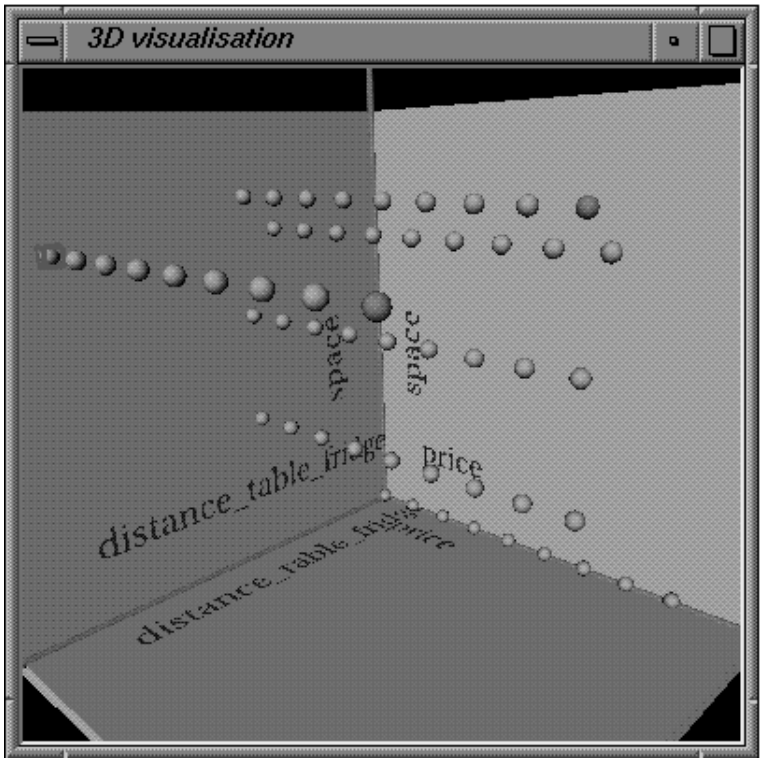

Figure 6.7: Comparaison de 3 critères, paréto-optimalité (Pareto 3D).

trois critères. Toutes les combinaisons sont donc représentées. Pour qu'une solution soit quatre-optimale, il ne doit pas y avoir de solution qui soit plus optimale qu'elle dans une des quatre trois-optimalités.

6.3.2.6 Balance

Nous avons reporté tout naturellement, dans un premier temps, le concept de pareto-optimalité á des dimensions plus élev ées en pensant avoir trouvé un bon moyen de représenter les solutions suivant n critères. Ce type de visualisation en trois ou en quatre dimensions est déj à assez difficiles à lire mais restent pertinent. Par contre pour davantage de dimensions, la visualisation risque de devenir surchargée (couleurs, formes, tailles, etc). Pour cette raison nous avons essayé de concevoir une visualisation en deux dimensions facile à lire et pouvant contenir un maximum de critères et d'informations pertinentes. Nous nous sommes en fait inspirés de la métaphore d'une balance romaine, à plateaux, qui nous semblait proche du marchandage et du processus qui consiste à faire des compromis. En français, on dit que l'on balance un objectif par rapport à un autre. Nous avons donc essayé d'imaginer une balance qui possède plus de deux plateaux. Notre visualisation indique la somme des critères en ordonnée et leur influence en abscisse:

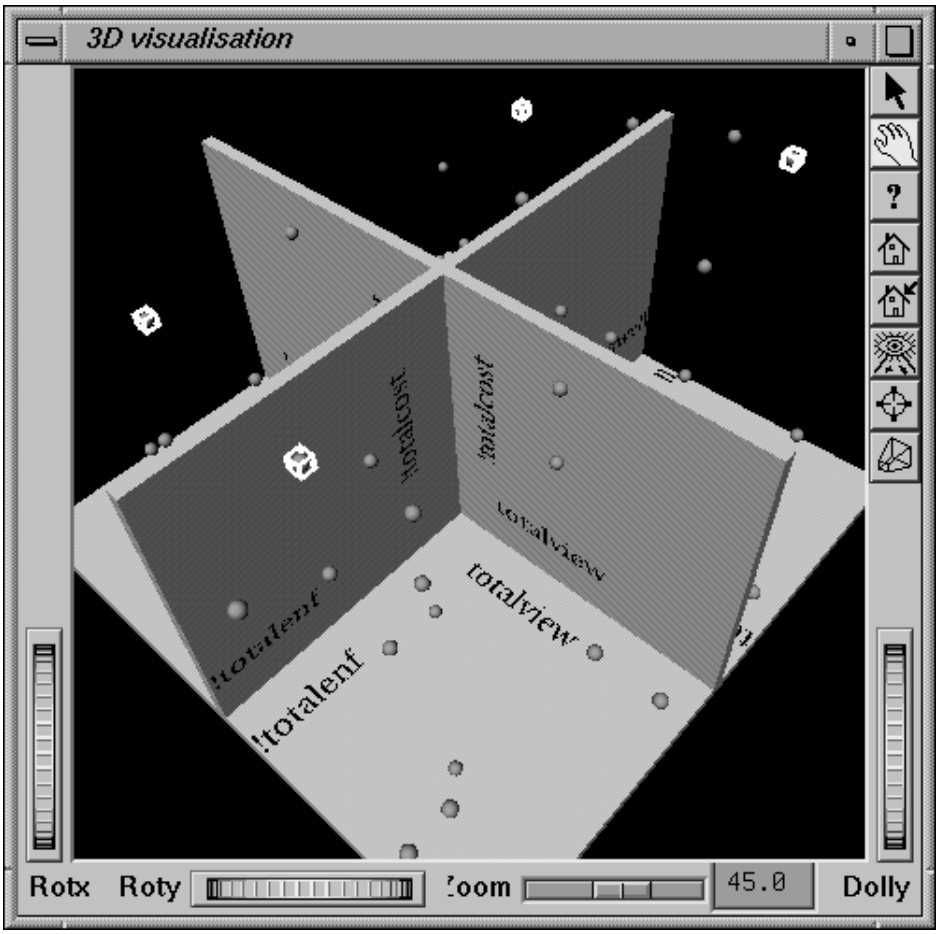

Figure 6.8: Comparaison de 4 critères, paréto-optimalité (Pareto 4D).

- plus une solution est haute, plus elle satisfait de critères en même temps.
- l'axe des abscisses se comporte comme une balance à plateaux. Si une solution se trouve plutôt à gauche, cela signifie que les critères situés à gauche sont plus forts. L'ordre des critères peut être changé par l'utilisateur, ce qui permet de donner un ordre d'importance à ces derniers.

Cette visualisation peut contenir autant de critères que l'utilisateur le souhaite et elle est facile à lire.

6.3.2.7 Interaction entre les visualisations

Toutes les visualisations sont connectées entre elles. Sélectionner une solution dans une des visualisations, indique automatiquement la même solution dans les autres visualisations. Il est souvent utile de voir un problème sous différents angles afin de pouvoir le comprendre et le résoudre.

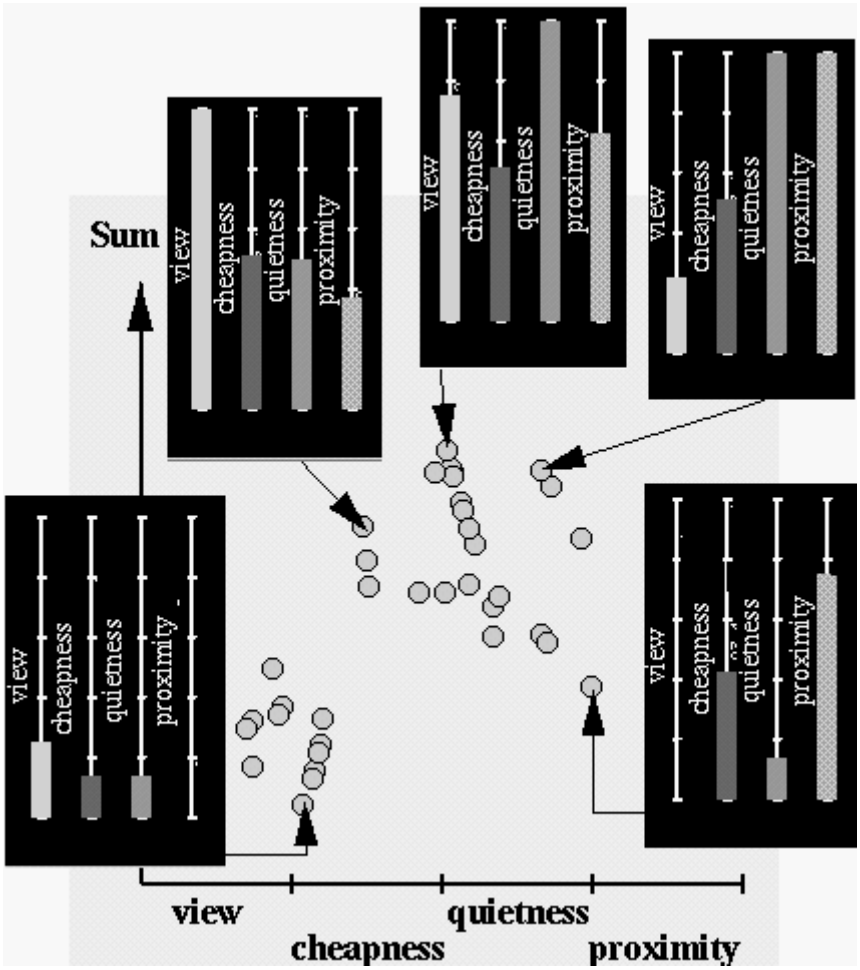

Figure 6.9: La visualisation Balance.

6.4 Plusieurs situations possibles

Plusieurs cas d'utilisation de l'assistant TradeOff sont imaginables. Nous verrons dans les chapitres finaux des scénarios d'utilisation de Comind qui illustrent ces cas. Par exemple, l'utilisateur peut se trouver dans les situations suivantes:

- le concepteur peut par exemple se rendre compte qu'aucune des solutions trouvées ne lui convient parce que celles qui l'intéressent ont été évitées. Le concepteur doit alors réviser la définition de ses contraintes en utilisant les outils adéquats pour trouver quelles sont les contraintes qui lui ont fait éviter les solutions qu'il désirait.

- le concepteur n'est pas satisfait par les solutions optimales selon ses critères. Par exemple, dans le cas d'une configuration de lotissement, les configurations les moins chères et les plus tolérables en terme de nuisance sonore ne sont pas

de bonnes solutions selon lui. Les maisons sont loin du lac, le cimetière est placé dans une région plutôt agréable , etc. Le concepteur réalise donc que la vue est un critère important dans son évaluation et retourne vers l'éditeur de l'assistant TradeOff pour définir ce nouveau critère (phénomène émergent voir Tomlin [Tomlin, 1986]). Remarquons ici l'importance que joue la visualisation pour la découverte de critères pertinents si l'on néglige la simplicité de l'exemple.

- quelques autres cas sont naturellement imaginables. Un cas très important se présente également lorsque, selon un ensemble de critères, le concepteur ne peut pas trouver les solutions optimales. Lorsqu'un critère est fort, un autre est mauvais, le concepteur se trouve face à un compromis à faire. C'est généralement dans de telles situations de conflit que les solutions les plus créatives sont découvertes.

6.5 Remarque

Souvent, dans une tâche de conception, l'utilisateur doit s'occuper, pour le même problème, des étapes sur et sous contraintes en même temps. Les processus de résolution de conflits et de production de compromis (≪TradeOff≫) ne sont pas toujours séparés pour l'utilisateur (6.10). Ils sont complémentaires et synergiques. Nous présentons l'assistant qui aide à traiter les conflits dans le chapitre suivant.

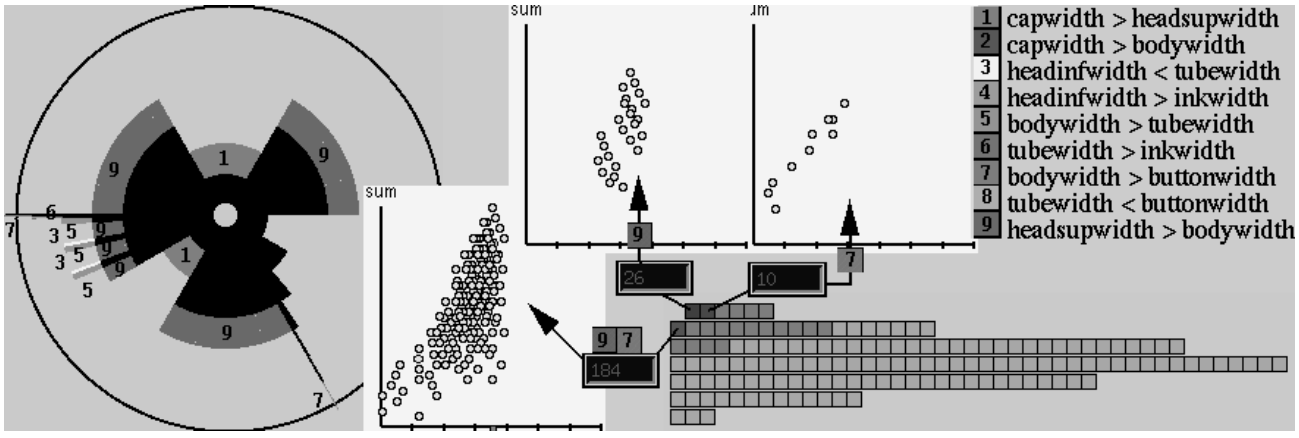

Figure 6.10: Le problème possède peu de solutions. Les assistants ElicitConflict et TradeOff permettent ensemble d'explorer l'espace potentiel.

Chapitre 7

Résoudre un problème sur-contraint

7.1 Résumé

Dans les problèmes sur-contraints et sur-déterminés, la relaxation de contraintes est une technique souvent utilisée pour libérer les problèmes et trouver des solutions. Cependant, trouver la contrainte appropriée n'est pas une tâche facile. Dans la conception, résoudre ce type de situation serait très utile car la résolution des conflits permet souvent de découvrir des solutions innovatrices. Néanmoins, la tâche est ardue pour le concepteur car les contraintes ne sont pas toujours relaxables et parce qu'il existe des conflits cachés dans la définition du problème. Conscients des limites des techniques de l'intelligence artificielle, nous proposons une méthode interactive pour la gestion des conflits dans la conception, prenant en compte les facultés créatives de l'utilisateur.

7.2 Introduction

Un problème sur-contraint ne possède aucune solution. Il est sur-contraint soit du fait d'une définition ≪conflictuelle≫ soit parce que le problème est sur-déterminé. Dans les problèmes ≪conflictuelles≫, il existe un sous-problème inconsistant, un k-conflit, qui empêche toute découverte de solution. Alors que dans les problèmes sur-déterminés, il n'y a pas de sous problèmes sur-contraints; le seul problème inconsistant est le problème lui-même. La cause provient alors souvent de la définition de trop nombreuses contraintes. Lorsqu'un problème est sur-déterminé, l'utilisateur doit relaxer des contraintes alors que lorsque la définition est conflictuelle, il doit repenser son problème. Nous allons présenter dans ce chapitre ces deux cas extrêmes et comment ils apparaissent. Dans la première partie de ce chapitre, nous présentons l'état de l'art des algorithmes permettant de résoudre des problèmes sur-contraints.

Dans la deuxième partie, nous présentons des outils et des visualisations que nous avons réalisé et leurs avantages. La troisième partie est une étude de cas permettant de comparer notre méthode avec les plus connues.

7.2.1 Etat de l'art

Lorsqu'un problème est sur-contraint, un ≪backtracking≫ classique (voir section 5.3.2, page 43) ne permet pas de trouver de solutions. La résolution de problèmes par satisfaction partielle de contraintes (Partial CSP ou PCSP) consiste à rechercher un sous-problème qui puisse être satisfait. Plus le sous-problème est grand, plus il est considéré comme proche du problème initial et plus il est supposé générer des solutions optimales. Différents critères d'optimisation, appelés métriques dans le domaine du PCSP, sont utilisés afin de décider quel est le plus petit ensemble de contraintes à relaxer, ou le plus grand sous-problème qui puisse être résolu. La satisfaisabilité maximale (Max CSP) est une de ces métriques. Elle permet de trouver les solutions qui satisfont le maximum de contraintes sans qu'il soit donné une priorité à l'une ou à l'autre. Le plus souvent, un mécanisme de pondération est utilisé afin de donner la priorité à la relaxation de certaines contraintes plutôt qu'à d'autres. Hiérarchie, probabilité, temporalité, ou dynamisme sont différentes méthodes permettant d'étendre la technique du Partial CSP ([Bakker and al., 1993], [Jampel, 1996b] et [Freuder and Wallace, 1996]). Selon le formalisme suivi par le Partial CSP, moins une valuation (une combinaison de paramètres, c'est-à-dire une solution potentielle) est violée, plus sa probabilité d'être une bonne solution est grande. Cependant, dans la conception, ce n'est pas toujours vrai. La méthode que nous proposons est différente car elle est basée sur l'intuition de l'utilisateur et parce qu'elle aide l'utilisateur à diagnostiquer les conflits. Notre méthode peut donc être vue comme une alternative au Partial CSP.

7.2.2 Violation de contraintes plutôt que conflit

Il est difficile de donner une définition de ce qu'est un conflit. Dans le contexte de la résolution de problèmes par satisfaction de contraintes (CSP), un conflit est une lutte entre plusieurs contraintes. Un ensemble de contraintes est en conflit avec un autre s'il ne peut pas être satisfait lorsque l'autre l'est. De façon à trouver tous les conflits parmi k contraintes, il est nécessaire de satisfaire la k-consistance. En fait, satisfaire une consistance de (k - 1) n'est pas suffisant afin de trouver tous les conflits parmi k contraintes.

Si l'on regarde le problème du point de vue du concepteur, lorsqu'aucune solution n'est trouvée, il veut ou bien trouver les solutions partielles les plus proches de la définition de son problème, ou bien trouver les conflits qui empêchent la découverte

de solutions afin d'être plus capable de redéfinir le problème. Trouver les antagonismes sémantiques qui empêchent la découverte de solutions est difficile pour une machine. De plus, un problème peut tout à fait ne produire aucune solution et ne posséder aucun réel conflit. Il semble donc qu'il soit difficile de raisonner avec les conflits. Dans le contexte de la conception, découvrir automatiquement les conflits les plus importants restera une tâche difficile à résoudre tant que les machines ne seront pas capables de distinguer les conflits sémantiques des autres. Parce que la découverte des conflits est une tâche ardue, nous proposons dans ce chapitre une approche différente, non automatique. L'idée sous-jacente principale consiste à proposer à l'utilisateur un ensemble d'outils qui puisse l'aider à raisonner sur un problème sur-contraint. Plus généralement, nous voulons montrer que lorsque les techniques de l'intelligence artificielle ne réussissent pas à résoudre un problème, les systèmes interactifs permettent de construire une intelligence augmentée, issue de la synergie entre l'intelligence humaine et l'intelligence artificielle.

De plus, de notre point de vue, les conflits sont juste une vue de l'esprit. Il n'y a pas de réels conflits, juste des violations de contraintes. Seul un ensemble de contraintes violé dans 100% des cas peut être considéré comme un conflit réel. Cependant, l'occurrence de ce cas est trop rare pour pouvoir être utilisé comme vocabulaire de base dans l'implémentation d'un algorithme de résolution de problèmes sur-contraints. Selon cette non croyance dans l'existence des conflits pour l'implémentation, nous proposons plusieurs visualisations basées sur la violation de contraintes plutôt que sur les conflits.

7.3 Aide à la relaxation de problèmes sur-contraints

L'entité d'aide à la découverte des conflits est distribuée en plusieurs sous-entités, chacune adaptée à un type particulier de problèmes. L'utilisateur est guidé dans son choix mais il reste toujours libre d'utiliser l'entité qu'il désire. Nous allons décrire plusieurs algorithmes qui représentent la contribution de l'intelligence artificielle à la résolution de problèmes sur-contraints. Les différentes visualisations, comme nous allons le voir, illustrent l'espace collaboratif entre l'intelligence artificielle et l'intelligence naturelle de l'humain. L'utilisateur pourra, afin de résoudre le problème, utiliser différents algorithmes comme s'il manipulait des outils au service de son intuition.

7.3.1 Distribution des responsabilités de violation

La première classe d'algorithmes est basée sur une distribution des ensembles de contraintes responsables de la non-satisfaction d'au moins une valuation. Pour chaque

combinaison de valeurs possibles dans l'espace, les ensembles de contraintes violées sont identifiés. A chaque fois qu'une solution est interdite, l'ensemble de contraintes violées correspondant est mémorisé. S'il est mis en mémoire pour la première fois, le nombre de valuations qu'il bride est mis à 1. Si ce n'est pas la première fois, ce nombre est incrémenté d'1. L'algorithme retourne les ensembles de contraintes bloquants ainsi que le nombre de valuations que chacun d'eux viole.

Un ensemble de contraintes Sijk=Ci, Cj, ..., Ck est considéré comme responsable de l'échec de la valuation x si et seulement si Ci, Cj, ..., et Ck sont les seules contraintes non satisfaites pour les valeurs contenues dans x. Pour un problème sur-contraint, la distribution des ensembles de contraintes responsables couvre donc tout l'espace des valuations de l'espace de recherche.

La première implémentation de cette idée est un algorithme très simple. Pour chaque valuation de l'espace de recherche, l'ensemble des contraintes violant correspondant est identifié. L'algorithme rend une carte de la topographie de l'espace des valuations organisé suivant les ensembles bloquants.

```
proc Generate-list-blocking-sets-of-constraints
   For each possible valuation
      set blocking-set empty
      For each constraint ci not satisfied
         append ci to the current blocking-set
         if blocking-clique in list-blocking-sets
         then
            add 1 to its number of occurrence
         else
            append blocking-set to list-blocking-sets with its counter set to 1.
```

7.3.1.1 Optimisation

Cet algorithme est mauvais en terme de complexité. Lorsque le nombre de paramètres et la taille de leur domaine augmentent, l'algorithme est très gourmand car son exécution est exhaustive. Nous proposons une optimisation de cet algorithme qui fonctionne mieux avec des problèmes de taille importante. L'algorithme consiste à trouver les ensembles de contraintes bloquants dans un ordre décroissant par rapport à leurs tailles. Le plus grand ensemble de contraintes bloquant est d'abord trouvé. Les valuations partielles correspondantes aux contraintes violées (≪no-goods≫) sont calculées. Toutes les valuations contenant ces ≪no-goods≫ sont éliminées de l'espace de recherche. Un ≪no-good≫ est une combinaison de k valeurs qui n'est présente dans aucune des solutions du problème. L'algorithme suivant est plus rapide que le précédent car il ne parcourt pas l'espace de manière exhaustive.

```
proc Generate-list-blocking-sets-of-constraints2
```

```
For each possible valuation
   For each constraint ci not satisfied
      append ci to the current blocking-set
      bigger-set-current <-- Find bigger set of constraints violated
                             containing those values
      set-occur <-- count and then remove all the valuations containing
                    this set of values
      if bigger-set-current in list-blocking-sets
        then
           add set-occur to its number of occurrence
        else
           append blocking-set to list-blocking-sets with its counter set to set-occur.
```

7.3.1.2 Deux types de problèmes sur-contraints: les visualiser et les relaxer

L'algorithme en lui-même serait inutile si aucune visualisation n'y était attachée. La visualisation lui donne une plus grande dimension en offrant une possible interaction avec l'utilisateur. Encore une fois, c'est parce que nous croyons au pouvoir de l'interaction homme-machine dans la résolution de problèmes que nous trouvons la visualisation utile pour augmenter aussi bien les facultés de l'utilisateur, que les performances des algorithmes.

La visualisation attachée à cet algorithme contient deux vues du même espace. Les deux représentent des ensembles de contraintes. Chaque ensemble bloque au moins une valuation. La somme de toutes les régions représente l'espace total des valuations dans le cas d'un problème sur-contraint. Il y a deux formes de visualisation possibles:

- **lattices**
  Les carrés, les ensembles de contraintes, sont ordonnés par leur nombre de contraintes (figure 7.1). Les ensembles les plus petits sont les premiers en haut. En effet, il est souvent plus facile de relaxer un petit ensemble qu'un plus grand, pour des raisons cognitives principalement. Chaque niveau du lattice a des ensembles qui possèdent le même nombre de contraintes. Le niveau x peut donc contenir les super-ensembles (≪super-set≫) des ensembles (≪set≫) du niveau x-1.

- **camembert**
  Les parts du camembert, qui correspondent aux ensembles de contraintes, sont ordonnées par leur nombre de contraintes (figure 7.2). Les ensembles les plus petits sont les premiers dans le sens contraire des aiguilles d'une montre avec l'origine à droite.

Dans les deux formes de visualisation, il y a deux vues du même espace. Ces espaces correspondent à **deux types de problèmes sur-contraints différents**:

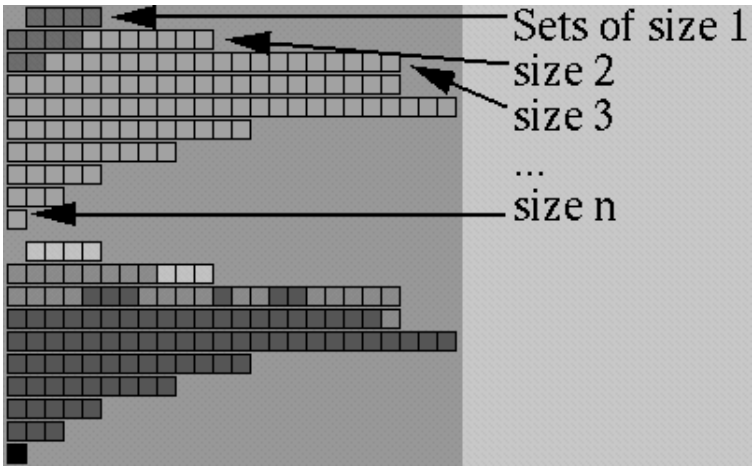

Figure 7.1: Distribution des responsabilités des violations sous forme de lattice. Chaque carré est un ensemble de contraintes qui interdit la découverte de solutions. Les deux vues représentent les mêmes ensembles de contraintes. Les carrés tous ensembles bloquent l'espace total des solutions. Dans la vue du dessus, en bleu, la relation entre un ensemble et ses sous-ensembles est dynamiquement indiquée. Dans la vue du dessous, en rouge, la relation d'intersection est dynamiquement illustrée; lorsque l'utilisateur clique sur un ensemble, les ensembles ayant au moins une contrainte commune sont indiqués (ils changent de couleur).

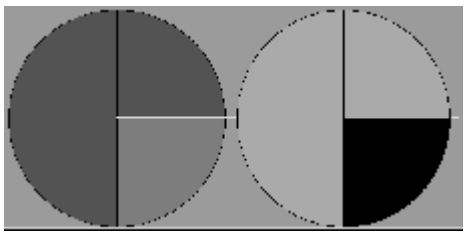

Figure 7.2: Une autre forme de distribution des responsabilités des violations

1. les problèmes conflictuels, où la taille du conflit est inférieure à la taille du problème, sont traités par l'espace du bas (respectivement de droite pour le camembert).

2. les problèmes sur-déterminés, où le plus petit conflit est le problème lui-même, sont traités par l'espace du haut (respectivement de gauche pour le camembert).

Le ≪trashing≫ (voir 5.3.4, page 44) est la cause de la non-découverte de solutions dans les deux types de problèmes. Ils sont tous deux inconsistants. Nous décrivons ci-dessous ces deux types de problèmes, comment les relaxer et les visualisations que nous proposons pour aider l'utilisateur à les soulager.

1. Dans la première famille de problèmes sur-contraints, les problèmes conflictuels, il existe un réel conflit plus petit que le problème.

   - Definition
     il existe un k-conflit, où $k < n$, n étant le nombre de contraintes contenues dans le problème.
   - Guide pour la relaxation
     La relaxation du problème correspond alors à la relaxation du plus petit k-conflit, le plus petit ensemble de contraintes en conflit. Nous parlerons, par la suite, de la notion de degré de violation exclusif. Ce degré correspond à la participation global de l'ensemble de contraintes sur le blocage du problème. Nous nous servons surtout de ce degré dans le cas où il vaut 100% afin de détecter les conflits. Remarquons que l'utilisateur peut utiliser aussi dans ce cas les outils disponibles pour la première classe de problèmes.
   - Visualisation
     L'espace du bas (respectivement de droite pour le camembert) représente aussi l'espace total des possibles. L'ordonnancement de l'espace précédent est conservé. Cependant, les ensembles sont coloriés en fonction d'un degré différent de violation. Une seule contrainte a besoin d'être violée afin que l'ensemble soit considéré comme responsable. Nous qualifions ce degré d'exclusif parce qu'il suffit qu'une contrainte de l'ensemble soit violée pour que l'ensemble soit considéré responsable: Ci ou Cj ou Ck ou etc. Lorsque l'ensemble est totalement noir, 100% de violation exclusive, aucune solution ne peut être trouvée sans relaxer cet ensemble. C'est un conflit. Lorsque l'utilisateur clique sur un ensemble, il obtient le nombre de solutions auxquelles participent les contraintes, ainsi que tous les ensembles qui possèdent au moins une des contraintes de l'ensemble sélectionné. Lorsque l'ensemble est un conflit, tous les ensembles sont mis en valeur parce qu'ils possèdent tous au moins une des contraintes du conflit.

2. Dans la deuxième famille de problèmes sur-contraints, les problèmes sur-déterminés, il n'y a pas de réel conflit mis à part le problème lui-même.

   - Definition
     il n'existe pas un k-conflit, où $k < n$, n étant le nombre de contraintes contenues dans le problème. Le seul k-conflit du problème est lorsque $k = n$.

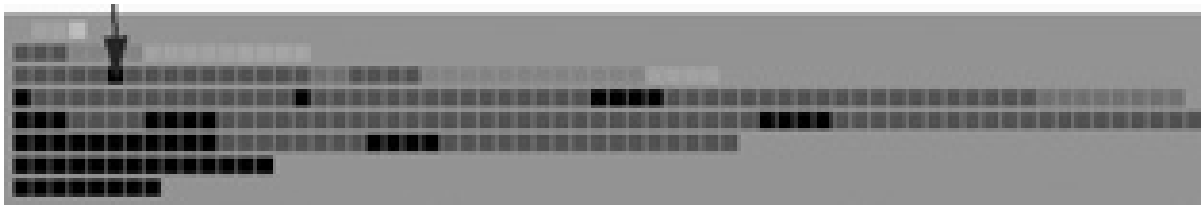

Figure 7.3: Les problèmes conflictuels (k-conflict). Il y a un k-conflit, c'est-à-dire un sous-problème sur-contraint (il y a un k-conflit tel que $k < n$, n le nombre maximum de contraintes. Pour relaxer le problème, il faut relaxer le carré noir le plus haut (le plus petit en nombre de contraintes).

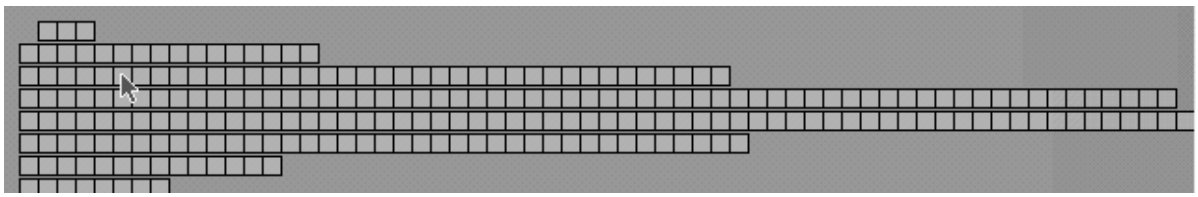

Figure 7.4: Les problèmes conflictuels (k-conflict). Un carré noir chevauche tous les autres carrés; ils partagent au moins une contrainte.

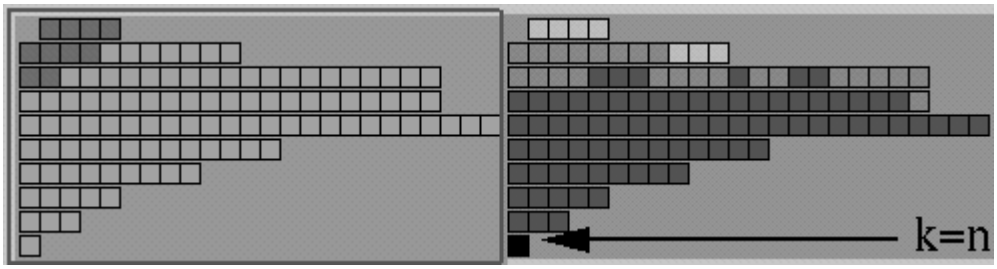

Figure 7.5: Les problèmes sur-déterminés. Le seul k-conflit est le problème lui-même (pas de k-conflit, $k < n$, n nombre maximum de contraintes). Pour relaxer ce problème, il faut prendre l'ensemble qui soit le plus petit et qui bloque le plus.

- Guide pour la relaxation
  Dans le cas où le problème est sur-déterminé, il faut raisonner sur les ensembles de contraintes les plus bloquants. Ces ensembles sont ceux qui génèrent le plus grand nombre de ≪no-goods≫ proportionnellement au nombre de valuations qu'ils traitent. Dans ce cas, il faut donc raisonner sur des valuations partielles. Par exemple, si l'on considère les variables x, y et z qui possèdent les domaines {1,2,3}, {1,2,3,4} et {1,2} :
  - La contrainte ≪x <> y≫ génère les ≪no-goods≫ (1,1), (2,2) et (3,3) alors que le nombre de valuations possibles est 3*4=12 (taille du domaine de x multiplié par la taille du domaine de y, les deux paramètres concernés par cette contrainte). Nous pouvons alors en

déduire un taux de blocage local de 3/12.

- Les contraintes $\ll x > y \gg$ et $\ll x == z \gg$ génèrent une seule valuation correcte: (2,1,2). Cet ensemble de contraintes génère donc $((3 * 4 * 2) - 1) = 23$ ≪no-goods≫ d'où l'on peut déduire le taux de blocage 23/24.

Il est alors évident que le deuxième ensemble de contraintes est davantage blocant. Nous introduirons ainsi par la suite la notion de degré de violation inclusif pour signifier ce taux de blocage. Remarquons aussi que, dans ce cas, l'utilisateur aura accès aux ≪no-goods≫ de chacun des ensembles de contraintes qui bloquent, grâce à la coopération entre l'assistant Conflict et l'assistant TradeOff (figure 6.10).

- Visualisation
  L'espace du haut (respectivement de gauche pour le camembert) utilise ce taux de blocage pour ordonner les ensembles. A l'intérieur de chaque famille ayant le même nombre de contraintes, les ensembles sont ordonnés par ce degré de violation. Toutes les contraintes de l'ensemble ont besoin d'être violées afin que l'ensemble soit considéré responsable. Nous qualifions ce degré d'inclusif car toutes les contraintes de l'ensemble sont violées: Ci et Cj et Ck et etc. Ainsi, chaque ensemble de contraintes est plus ou moins sombre suivant la valeur de ce pourcentage de violation inclusif. Plus il est sombre, plus il est bloquant proportionnellement à l'espace qu'il couvre. Remarquons que dans le cas des lattices, au contraire de la visualisation sous forme de camembert, la taille des carrés est toujours la même. Nous n'avons pas représenté ici le nombre de solutions bloquées par un ensemble. Il faut cliquer sur l'ensemble pour voir le nombre de solutions qu'il bloque, ainsi que pour visualiser tous ses sous ensembles (qu'il faut prendre en compte pour savoir combien de solutions un ensemble bloque). Si l'ensemble de contraintes est effacé du problème, le nombre de solutions correspondra exactement au nombre de solutions que cet ensemble viole plus le nombre de solutions que ses sous-ensembles violent (voir 7.6 et 7.7).

Par exemple, les figures 7.2 illustre la découverte des conflits dans le problème du coloriage d'une carte. Trois pays (La France, la Suisse et l'Italie) peuvent être coloriés soit en bleu soit en rouge. Des pays limitrophes ne peuvent pas être de la même couleur. A gauche de la figure 7.2, il est possible de voir que seul un ensemble de contraintes est un conflit (violé 100% du temps au sens exclusif). Il s'agit de l'ensemble contenant les trois contraintes du problème (couleurFrance ! = couleurSuisse, couleurFrance ! = couleurItalie, couleurSuisse ! = couleurItalie). Parce que les trois contraintes binaires toutes seules violent proportionnellement à

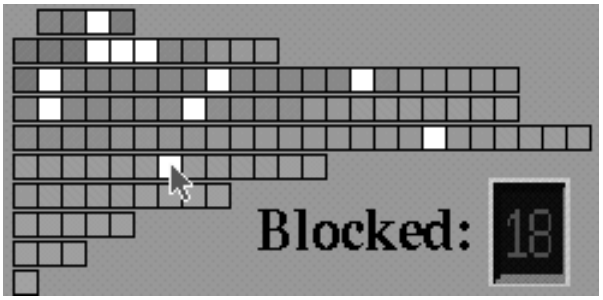

Figure 7.6: Dans les problèmes sur-déterminés, la relation entre un ensemble et ses sous-ensembles est dynamiquement indiquée.

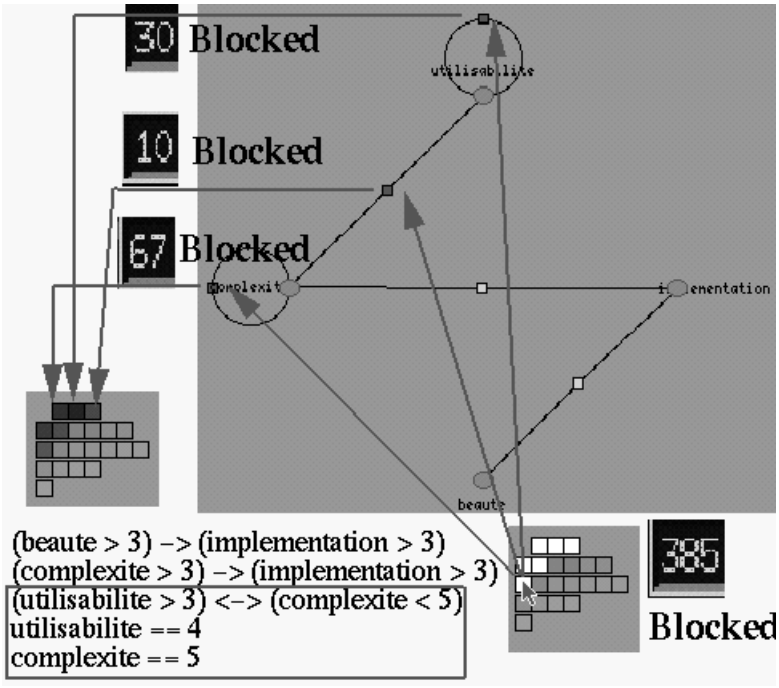

Figure 7.7: L'asistant ConflictElicit est lié avec l'assistant Param-Def, de sorte que lorsque l'utilisateur clique sur un ensemble de contraintes qui bloque des solutions, cet ensemble et ses sous-ensembles sontt mis en valeur dans le Param-Def.

leur couverture plus de solutions, leur degré inclusif de violation est plus grand que l'ensemble de contraintes possédant les trois. Pour cette raison, les trois contraintes binaires sont plus foncées (50% de degré inclusif) que leur combinaison (25% de degré inclusif). La visualisation en dessous illustre les degrés exclusifs de chaque contrainte (100% pour la contraintes combinant les trois et 50% pour les autres individuellement). Seule la contrainte combinant les trois règles binaires est un conflit.

7.3.2 Elagage de l'espace de recherche basé sur l'évaluation de Knuth

Lorsque le nombre de paramètres et la taille de leur domaine augmentent, la première famille d'algorithmes, basée sur la distribution des responsabilités des violations, est de complexité exponentielle et nécessite donc un calcul trop coûteux. Pour cette raison, nous proposons une approche interactive qui combine différents algorithmes simples afin d'élaguer l'espace initial du problème sur-contraint en ce concentrant sur le sous-espace le plus bloquant et afin de rendre l'utilisation de la classe d'algorithmes précédant possible. Cette approche utilise principalement l'algorithme de Knuth pour évaluer l'efficacité d'un programme de backtracking [Knuth, 1975]. L'algorithme de Knuth utilise une approche probabilistique basée sur l'exploration aléatoire de l'arbre de recherche. Pour chaque solution partielle, une continuation valide est aléatoirement choisie. Quand l'algorithme atteint une feuille de l'arbre, le nombre estimé de solutions est retourné, en fonction du nombre d'embranchements valides écartés. L'algorithme parcourt uniquement un noeud à chaque niveau de l'arbre de recherche et est donc très rapide.

Nous utilisons l'algorithme de Knuth afin d'estimer le nombre de solutions de chaque sous-problème, c'est-à-dire le problème initial auquel est enlevé chaque paramètre ou chaque couple de paramètres:

```
For each parameter pi in P
estimate with Knuth's algorithm the number of solution of P \ pi

For each parameter pi in P
    For each parameter pj in P, j>i,
estimate with Knuth's algorithm the number of solution of P \ {pi, pj}
```

La visualisation attachée à cet algorithme est un simple histogramme. Elle représente chacune des sous-régions du problème initial auquel a été enlevé un ou deux paramètres et l'estimation de son nombre de solutions.

Lorsqu'un paramètre est enlevé du problème initial, ainsi que les contraintes contenant ce paramètre, l'histogramme indique le nombre de solutions que possède le problème ainsi modifié.

Lorsqu'un couple de paramètres a été enlevé, l'histogramme indique le nombre de solutions que possède le problème sans ces paramètres ainsi que le nombre de solutions sans chacun des deux paramètres. Il est ainsi possible de savoir si les contraintes les plus bloquantes sont unaires ou binaires et quels sont les paramètres responsables. Ainsi, l'utilisateur peut choisir le sous-espace interdisant le plus de solutions afin de le relaxer grâce à la première famille d'algorithmes.

Cette algorithme est très rapide. Cependant du fait que ce n'est qu'une estimation, il n'est pas très précis. Afin de réduire le facteur aléatoire, plusieurs estimations du même sous-problème sont faites puis la moyenne est retournée. Afin d'augmenter la précision, il est possible d'augmenter le nombre d'estimations. C'est un compromis à faire entre la rapidité et la précision. Il est aussi possible d'imaginer d'enlever du problème initial des ensembles de paramètres plus grands afin de détecter des inconsistances plus importantes. Encore une fois c'est un compromis entre rapidité et précision. De plus, l'objectif principal est de guider l'utilisateur dans son choix de la région à réviser et ainsi d'élaguer l'espace initial pour une future recherche plus précise.

D'autres méthodes comme celle-ci auraient pu être conçues basées sur l'évaluation de Knuth; il aurait été possible par exemple d'enlever chaque contrainte une à une. Cependant le nombre d'évaluations aurait été ainsi plus important et l'algorithme moins rapide.

7.3.3 Un calcul plus analytique

Afin de ne pas traverser tout l'espace des possibles, l'utilisateur peut sélectionner à l'aide de la souris un ensemble de contraintes et obtenir ses degrés de violation inclusifs et exclusifs respectifs. Ainsi, l'utilisateur peut savoir presque immédiatement combien de solutions un ensemble de contraintes spécifique restreint. L'algorithme utilisé est pratiquement analytique. L'ensemble de contraintes est transformé dans sa forme matricielle si deux paramètres sont concernés, une forme circulaire si plus. Le pourcentage est alors le nombre de sous-valuations non satisfaites, contenant une valeur concernée par cet ensemble de contraintes, sur le nombre total de sous-valuations.

7.3.4 Problèmes ≪normaux≫: l'assistant Solve

L'assistant Solve utilise différents algorithmes de recherche ayant différents degrés d'efficacité et de rapidité. Il fournit aussi différentes visualisations de l'espace de recherche (voir chapitre 5, page 41). C'est un espace interactif pour la résolution de problèmes possédant un nombre raisonnable de solutions.

7.3.5 Performances

Nous avons testé les algorithmes sur plusieurs types de problèmes. Nous avons finalement obtenu les résultats suivants:

- la première classe d'algorithmes, qui permet de classifier les responsabilités des violations de contraintes, requiert dans le pire des cas n^p itérations avec n la taille moyenne des domaines du problème et p le nombre de paramètres.
- le pire cas est atteint lorsque chaque solution interdite correspond à un différent nogood (combinaison de valeurs bloquante). Heureusement, dans la plupart des cas, l'espace d'échec est distribué en un nombre raisonnable de nogoods, bien inférieur à n^p, et donc l'algorithme est plus rapide.
- un bon problème pour cette famille d'algorithmes est un problème où les contraintes ne sont pas toutes inter-connectées et qui possédent différentes zones de blocage.
- en général, $n^p < 10000$ est la garantie d'une calculation rapide (environ 5 secondes sur une Silicon Graphics Indy avec les algorithmes programmés en Allegro common Lisp).
- la seconde classe d'algorithmes est bien meilleure pour un grand problème et en général pour avoir une estimation de la distribution des conflits dans le problème. Néanmoins cet algorithme est bien moins précis.

7.4 Etude de cas

Afin d'illustrer ces algorithmes dans le contexte de problèmes réels, nous présentons dans la section suivante des scénarios dans lesquels nous illustrons les cas extrêmes que le concepteur peut rencontrer.

7.4.1 Les problèmes sur-déterminés

Un problème sur-déterminé est un problème qui possède des contraintes redondantes s'entremêlant. La définition du problème n'est pas minimale et certaines contraintes peuvent être relaxées ou fixées et certains domaines peuvent être agrandis. Par exemple, nous illustrons ce type de problèmes sur-contraints avec l'étude de cas suivante [Jampel, 1996b]:

Michael, John and Alan must attend work-sessions taking place over four half-days. John and Alan want to present their work to Michael and Michael wants to present his work to John and Alan. Let Ma, Mj, Am, Jm be the four presentations (Michael to Alan, Michael to John, ...). Those variables have the domain $1, 2, 3, 4$ *(the four half-days). Each presentation takes one half-day. Michael wants to know what John and Alan have done before presenting his work. Michael would like not to come the fourth half-day and Michael does not want to present his work to Alan and John at*

the same time. Someone who attends a presentation cannot present something in the same half-day. Two different persons cannot present to the same person at the same time.

Nous introduisons donc les contraintes suivantes dans l'assistant Param-Def (figure 7.8), l'assistant de définition de problèmes de COMIND:

- $C0 : Ma > Am$, $C1 : Ma > Jm$, $C2 : Mj > Am$ and $C3 : Mj > Jm$.
  parce que Michael veut savoir ce qu'ont fait Alan et John avant de présenter son travail. Le signe supérieur signifie ≪doit avoir lieu avant≫.

- $C4 : Ma <> 4$, $C5 : Mj <> 4$, $C6 : Am <> 4$ and $C7 : Jm <> 4$.
  parce que Michael ne veut pas venir à la quatrième session.

- $C8 : Ma <> Mj$.
  car Michael ne veut pas présenter son travail à Alan et à John en même temps.

- $C9 : Ma <> Am$, $C10 : Ma <> Jm$, $C11 : Mj <> Am$, $C12 : Mj <> Jm$ and $C13 : Am <> Jm$.
  parce qu'une personne qui présente son travail ne peut pas suivre une présentation en même temps et parce que deux différentes personnes ne peuvent pas présenter leur travail à la même personne en même temps.

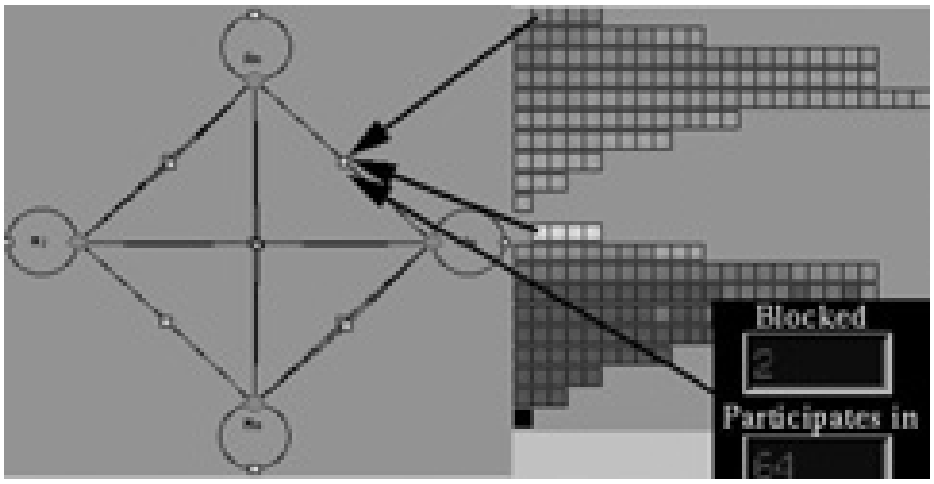

Figure 7.8: Définition du problème et distribution des responsabilités des violations. Le graphe correspondant au problème est complet. Le seul k-conflit est le problème lui-même. C'est donc un problème sur-déterminé.

L'algorithme d'évaluation de Knuth estime que le nombre de solution est nul. Le problème est sur-contraint. Le problème contient quatre paramètres contenant chacun quatre valeurs. Il y a donc $4^4 = 256$ valuations (solutions potentielles) dans

l'espace du problème. 256 est bien inférieur à 10000 et donc la distribution des responsabilités des contraintes est très vite calculée, en moins d'une seconde. Comme il est possible de le constater sur la figure 7.8, le problème est complet; le graphe, représentant le problème, est totalement inter-connecté. Nous somme donc ici dans le pire cas pour notre algorithme.

Selon la visualisation donnée par la distribution des ensembles de contraintes responsables de violation (figure 7.8), le seul k-conflit est l'ensemble contenant toutes les contraintes du problème. Il n'y a pas de k-conflit plus petit que le problème lui-même. Nous sommes donc bien dans le cas sur-déterminé. L'utilisateur doit utiliser la visualisation en haut de l'écran traitant le degré inclusif de violation. Cela va l'aider à trouver quel est l'ensemble de contraintes préférable à relaxer. L'utilisateur veut dans ce cas relaxer le plus petit ensemble de contraintes, le plus proche de la définition initiale du problème. Il est bien évident que l'intervention de l'utilisateur est essentielle dans ce processus car il est le seul capable de juger de la sémantique des contraintes, et le seul apte à juger de la facilité à relaxer un ensemble plutôt qu'un autre. Cependant la machine par le biais de la visualisation est d'une aide importante pour l'aider à se concentrer sur les bons ensembles de contraintes. La figure 7.8 indique que si l'utilisateur supprime la contrainte 13, 8, 5 ou 4, il est assuré de trouver deux solutions. Ce résultat valide l'exactitude de notre système puisque les mêmes résultats avaient été trouvés lors de l'atelier sur les systèmes sur-contraints [Jussien and Boizumault, 1996]. Cependant, l'intérêt de notre approche est évidente car elle laisse l'utilisateur libre de choisir lui-même les contraintes qu'il veut relaxer et car la visualisation rend le travail de la machine transparent aux yeux de l'utilisateur. Ainsi, l'utilisateur peut choisir de relaxer un plus grand ensemble s'il désire avoir plus de solutions et les comparer ensuite avec l'assistant TradeOff. Aucun algorithme automatique ne permet cette flexibilté et cette coopération entre l'humain et la machine.

7.4.2 Les problèmes conflictuels

Nous allons maintenant voir un exemple de conception où la définition est conflictuelle . Un problème conflictuel est un problème où il y a au moins un réel conflit (un degré de violation exclusif à 100%). L'utilisateur est alors dans l'obligation de relaxer ces réels conflits afin de résoudre son problème. Dans ce scénario, l'utilisateur doit concevoir une montre possédant un écran pouvant recevoir des images animées, pouvant servir de visio-phone portable par exemple. L'utilisateur défini les contraintes suivantes dans l'assistant Param-Def:

1. The size of the watch screen has to be 4 times smaller than the resolution
2. The material of the bracelet is plastic if and only if the flexibility is good.
3. The size of the watch is the same than the weight if and only if the material of the bracelet is wood.
4. The size of the watch screen has to be equal to 20 mm.
5. If the resolution of the screen equal 240, the price is greater than 5000.
6. If the color of the watch is gold, the material of the bracelet has to be steel.
7. If the color of the watch is silver-grey, the material of the bracelet has to be steel.
8. The flexibility of the bracelet has to be good.
9. The color of the bracelet has to be different than green.
10. The color of the bracelet has to be different than red.
11. If the price equal 10000, the color of the bracelet is blue.
12. The material of the bracelet has to be different than plastic.

Le concepteur utilise l'algorithme de Knuth pour évaluer le nombre de solutions. Il n'y en a aucune. Le problème contient 7 paramètres: le prix qui possède le domaine {1000 2000 10000}, la couleur {beige blue red green gold silver-grey}, le matériel {wood plastic steel}, la taille {20 30 40}, la résolution de l'écran {40 80 240}, la flexibilité du bracelet {good bad}, et le poids {40 80 120}. La taille de leur domaine respectif permet de calculer le nombre maximum de solutions: $3*6*3*3*3*2*3 = 2916 < 10000$. L'algorithme, permettant d'obtenir une visualisation de la distribution des contraintes suivant leurs responsabilités de violation est donc utilisable.

La visualisation du haut de l'écran nous indique que les contraintes 2, 8 et 12 ont le plus grand degré de violation inclusif. Celle du bas indique que le plus petit ensemble en conflit (100% degré exclusif) est celui possédant les contraintes 2, 8 et 12 (figure 7.9). On peut donc en conclure que tous les ensembles bloquants possèdent au moins une des ces trois contraintes. Aucune solution ne peut donc être trouvée sans que ne soit relaxé ce conflit.

Afin de visualiser plus en détail ce conflit, un autre type de visualisation est proposé à ce niveau. Lorsque l'utilisateur désire voir plus en détail des contraintes et leur combinaison, il peut les sélectionner à l'aide de la souris et obtenir une visualisation de leur forme matricielle. Cette visualisation permet de voir les différentes valeurs interdites par une combinaison de contraintes. Par exemple, la figure 7.10 montre que la combinaison des contraintes 2, 8 et 12 interdit toutes les combinaisons de valeurs. L'utilisateur peut ainsi choisir d'ajouter une valeur au domaine d'un paramètre ou modifier une contrainte afin que le conflit soit relaxé. Il peut pour

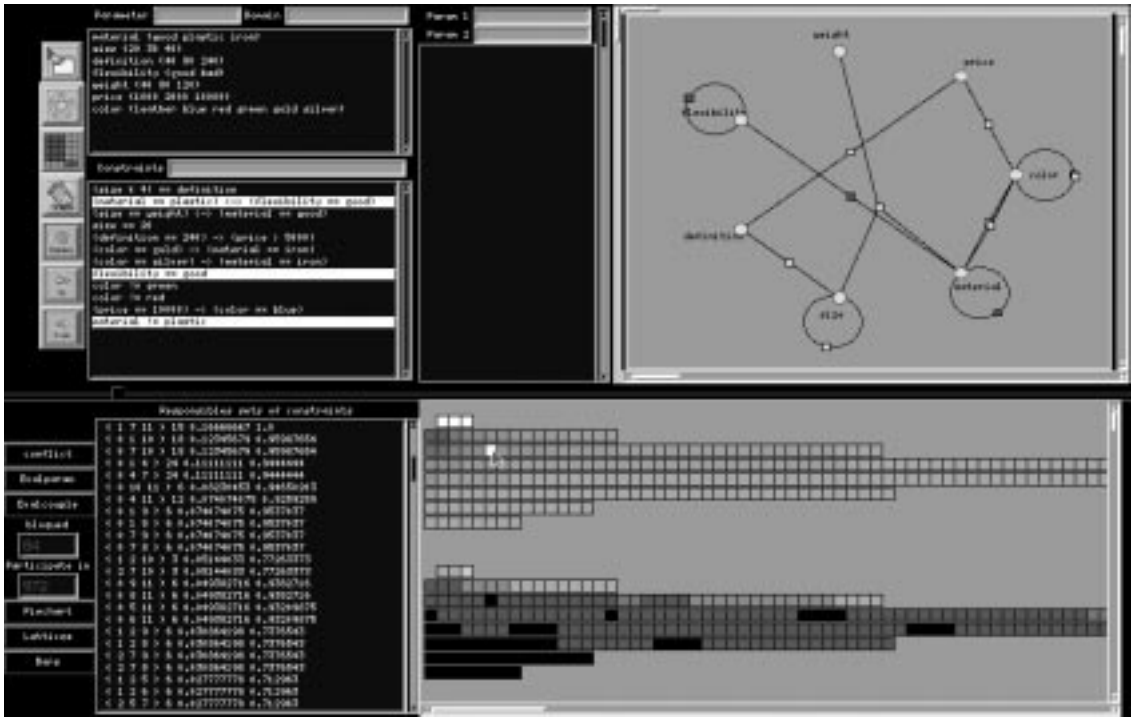

Figure 7.9: Le plus petit ensemble en conflit est choisi, apparait alors le nombre de solutions bloquées par l'ensemble ainsi que le nombre de valuations qu'au moins une des contraintes de l'ensemble ne satisfait pas.

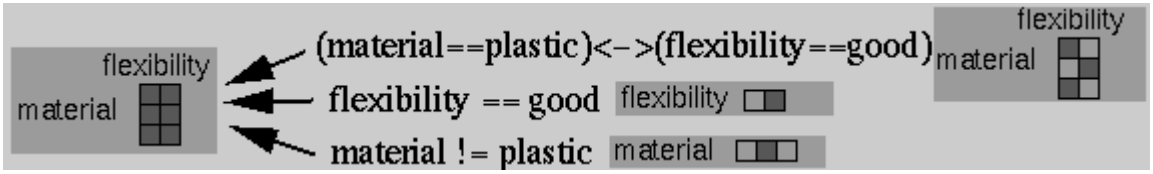

Figure 7.10: Formes matricielles des contraintes responsables du conflit

cela directement interagir avec la forme matricielle de chaque contrainte (puisque toutes les contraintes sont aussi bien définie sous forme matricielle que logique). Ainsi en interagissant graphiquement, il peut trouver une relaxation qui le satisfait et résoudre ainsi le problème qui était sur-contraint. Remarquons aussi, comme dans ce cas précis, que lorsque le conflit est d'ordre sémantique, l'utilisateur peut être amené à repenser sa définition, la machine servant ainsi de système critique. Il est en effet évident, lorsque l'on lit la définition des contraintes 2, 8 et 12, qu'elles ne peuvent pas être satisfaites en même temps. Le concepteur va donc être amené à repenser son problème en considérant par exemple un nouveau matériel aussi flexible que le plastique mais plus esthétique. Ou même, cas extrême, il sera amené à supprimer la contrainte 12.

7.4.3 Minimiser la taille du problème

Dans le cas où $n^p > 10000$, la première classe d'algorithmes requiert un temps de calcul trop long (plus de 5 secondes sur une Silicon Graphics Indy avec les algorithmes programmés en Allegro common Lisp). Dans cet exemple, nous allons illustrer l'utilité de l'algorithme d'estimation de Knuth. Nous traitons ici un cas de conception grand ne pouvant pas être résolu par la première forme d'algorithmes. Nous considérons un problème de conception inspiré du travail de Navinchandra [Navinchandra, 1991]. Une région au Sud-Ouest d'une ville doit être développée. Huits lots de terrain sont disponibles pour de futurs développements. Six différentes sections sont à développer: une aire de récréation, un immeuble, un lotissement de 50 villas, une grande décharge, un cimetière, une école et un supermarché. Un plan du site est représenté dans ShowCase, un des assistants de COMIND permettant de stocker des documents multimedias de conception (figure 9.1, page 94). Le plan présente huit terrains disponibles pour un futur développement. Les lots 3, 5, 7 et 9 sont des sites relativement plats avec un sol assez bon. Les lots 10 et 12 sont un peu en pente dans une jolie région boisée. Le problème consiste à trouver la meilleure assignation de lots aux différents types de développement. Une conception est complète lorsqu'à chaque type de développement (immeuble, maisons,...) est assigné un terrain. La conception finale doit satisfaire l'ensemble de contraintes suivantes:

- la décharge publique ne doit ni être visible depuis les maisons, ni depuis l'immeuble.
- le supermarché ne doit ni être visible depuis les maisons, ni depuis l'immeuble, ni depuis la décharge et ni depuis le cimetière.
- les terrains en pente doivent être évités pour les fondations.
- les sols instables doivent être évitées pour les constructions qui nécessitent des fondations.

Aucune solution n'est trouvée. Le problème est sur-contraint. Le concepteur doit donc trouver les conflits implicites dans la définition du problème. Parce que le problème est trop grand pour utiliser la première classe d'algorithme ($n^p > 10000$), et parce qu'il est difficile pour un humain de raisonner sur plusieurs contraintes et paramètres à la fois, l'assistant ConflictElicit propose à l'utilisateur d'élaguer l'espace de recherche afin de découvrir plus facilement les conflits. Un simple histogramme basé sur l'évaluation de Knuth, est proposée à l'utilisateur (figure 7.11). Comme il est possible de le voir sur cette visualisation les sous-problèmes ne possédant ni les appartements ni les maisons sont parmi ceux qui génèrent le plus

de solutions. Les contraintes concernant les maisons et les appartements sont donc parmi les plus bloquantes. L'utilisateur se concentre donc maintenant sur le sous-

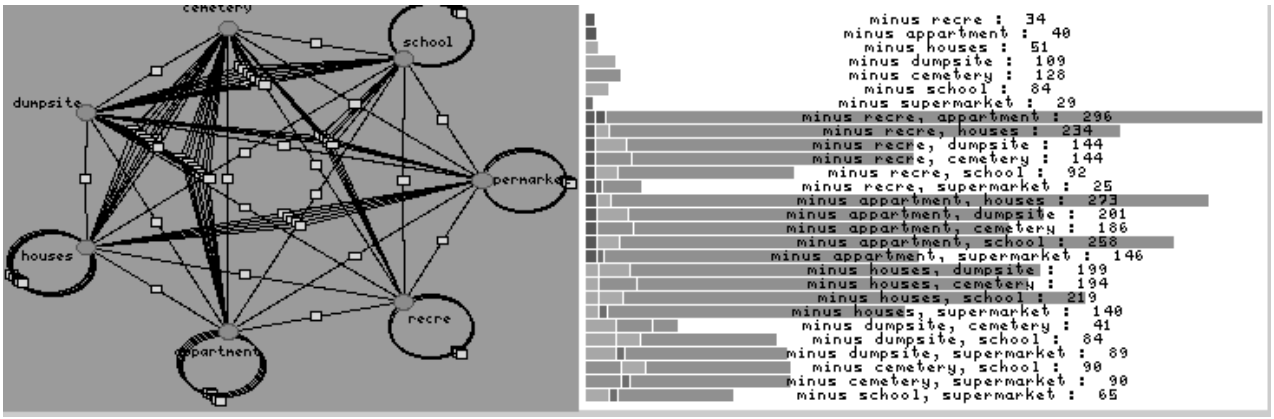

Figure 7.11: Elagation de la taille du problème, un sous-assistant de l'assistant Elicit-Conflict de Comind. Chaque barre de;'histogramme représente le nombre de solutions que possède le problème initial auquel a été enlevé respectivement un paramètre ou un couple de paramètre.

problème composé de ces deux paramètres. Le problème étant bien plus petit et sa taille maintenant bien inférieure à 10000 (valeur subjective dépendante de la plate-forme et du langage utilisés), l'utilisateur peut facilement obtenir la distribution des responsabilités des violations (voir la figure 7.12). Il constate qu'une des contraintes les plus gênantes est celle qui interdit les appartements d'être sur le lot 3. En effet, le lot 3 est celui qui est adjacent à l'autoroute. Si l'utilisateur ne voulait pas, par exemple, insonoriser ces appartements, il pourrait se concentrer sur une autre contrainte. Parce que le supermarché ne peut pas faire face à la majorité des constructions, l'utilisateur aurait pu aussi décider de remplacer le supermarché par une épicerie plus discrète et s'adaptant mieux au paysage. Finalement, si l'utilisateur enlève la contrainte interdisant les appartements sur le lot 3 de sa définition, il obtient 16 solutions (voir la figure 7.13).

7.5 Conclusion

L'étude des conflits est un concept difficile. Nous avons montré dans ce chapitre comment résoudre une problématique compliquée en utilisant les facultés de calcul des machines et en intégrant la créativité humaine via l'interaction à travers la visualisation. Nous avons aussi présenté une alternative du Partial CSP pour résoudre les problèmes sur-contraints. Notre approche est basée sur les techniques de l'intelligence artificielle et sur une collaboration asynchrone entre l'humain et la machine. L'objectif principal est de faire intervenir l'intuition humaine dans un

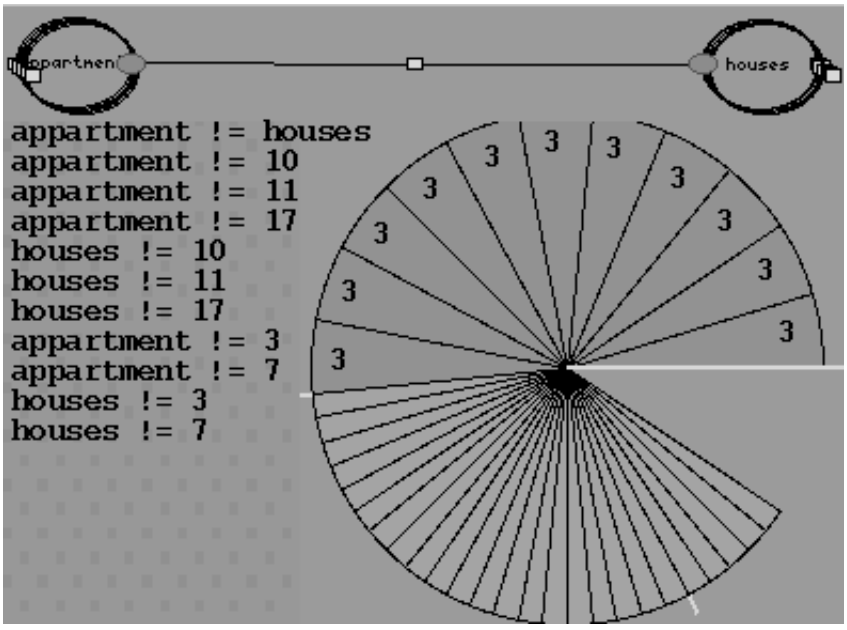

Figure 7.12: Distribution des responsabilités du sous-problème. Il n'y a pas de conflit. Les contraintes du sous-problème bloquent le même nombre de solutions partielles.

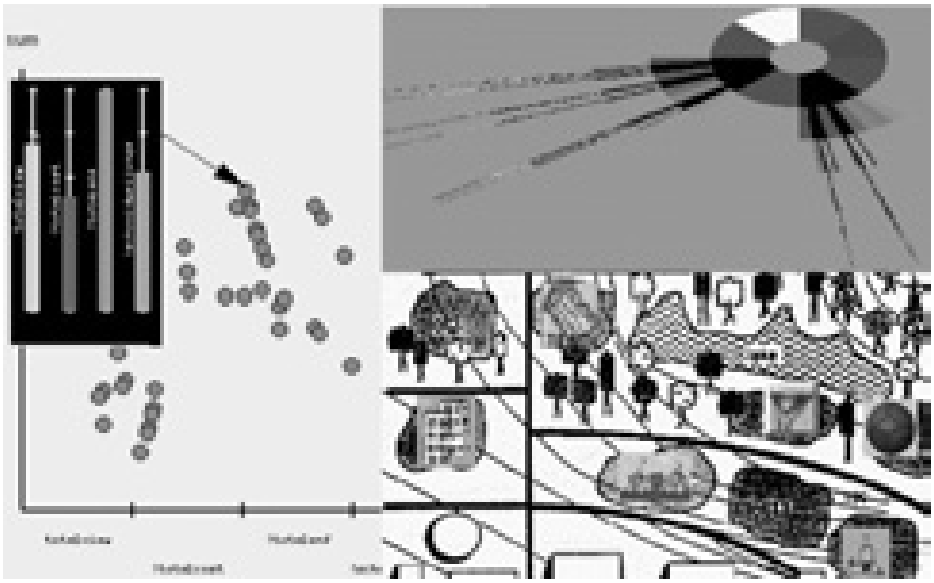

Figure 7.13: Le problème finalement libéré

processus de recherche interactif.

Chapitre 8

Les assistants réflexifs

Chacun des assistants de calcul, que nous avons présenté dans les chapitres précédents, possède quelques assistants réflexifs qui permettent de communiquer avec l'utilisateur et de refléter leur travail en commun. Nous présentons dans ce chapitre les deux assistants réflexifs qui interviennent sur le processus global de conception: l'assistant History, qui permet de soutenir le raisonnement du concepteur et l'assistant ShowCase, un système multimédia dédié à la conception.

8.1 Soutenir le parcours du concepteur

8.1.1 Quelques notions

De quelle façon aider une personne qui veut créer, découvrir, organiser consciemment ses connaissances? Est-ce la protection contre une explosion d'activités de pensée qui l'aidera à trouver de nouvelles voies de raisonnement, ou le contraire? Faut-il l'aider à auto-réguler ses réflexions ou faut-il, au contraire, l'aider à créer le chaos dans son raisonnement? Afin de trouver les réponses à ces questions, différentes systèmes de régulation de la pensée vont être présentées dans ce chapitre. Ces mécanismes régulateurs ont, en effet, une place très importante dans le processus de conception. Nous allons essayer de voir jusqu'à quel point ils peuvent brider un tel processus de création. Notre recherche bibliographique nous a permis de trouver un consensus existant entre plusieurs disciplines sur l'existence d'un méta-niveau régulateur. Nous ne présentons dans cette thèse que les travaux des psycho-cogniticiens (pour en savoir davantage sur cette réflexion et sur les autres approches, voir [Lalanne, 1994]).

8.1.1.1 Attention et contrôle en mémoire

Nous présentons, dans cette section, comment les psycho-cogniticiens modélisent le contrôle de l'activité en mémoire. Ainsi, il sera plus facile de comprendre comment il est possible de le minimiser ou de le renforcer pour aider l'expert dans sa tâche

de conception.

On considère actuellement que trois composants au moins composent la mémoire de travail: deux systèmes esclaves (la boucle articulatoire et le calepin visuo-spatial) sous le contrôle d'un troisième composant, l'administrateur central [Baddeley, 1990]. Malheureusement, les recherches sur la mémoire de travail se sont plus souvent concentrées sur les systèmes auxiliaires, plus abordables, que sur l'administrateur central. Ce dernier fonctionnant davantage comme un système attentionnel que comme une unité de stockage mnésique, une étude des théories de l'attention s'est avérée essentielle.

- **L'étude de l'attention**

  - **Vigilance**
    Les tâches, nécessitant une certaine forme de stockage à court terme, donnent lieu à une chute de la vigilance au cours du temps. Par exemple, écouter un flot continu de nombres et essayer de détecter trois chiffres impairs successifs est une de ces tâches. Alléger la mémoire de travail de l'expert pourrait donc être une bonne solution pour améliorer sa vigilance dans son processus de conception. L'état du problème de conception, dans l'assistant Param-Def (voir chapitre 4, page 29) servira de mémoire temporaire et soulagera ainsi l'empan mnésique de l'expert.

  - **Performance en situation de double tâche**
    En général, les sujets, qui ont deux tâches mnésiques à réaliser, ont des difficultés à les mener toutes les deux de front. Ils ont tendance, si cela est nécessaire, à en mettre une en attente jusqu'à que l'autre soit terminée. Posner et Boies (cités par [Baddeley, 1990]) émettent l'hypothèse que plus le stimulus d'une tâche secondaire est chargé attentionnellement, plus grande sera la tendance à différer la réaction à celui-ci dans le but de mener à bien la tâche principale qui nécessite une grande attention. Dans le cas du processus de conception, ce type de régulation risque de provoquer l'oubli d'idées. En effet, une voie de raisonnement, que l'administrateur central aura jugé secondaire, sera mise en attente pour soulager l'avancement de la voie principale. Un mauvais jugement de l'administrateur central signifiera alors la perte d'une bonne idée. Pour éviter cela, nous allons chercher à bâtir une structure qui conserve le cheminement de la quête du concepteur et donc toutes ses voies de raisonnement afin qu'il ne soit pas victime de ses processus régulateurs de pensée.

- **Automaticité**

  La plupart d'entre nous sommes capables de réaliser des tâches complexes telles que conduire tout en parlant avec relativement peu d'interférences, bien que l'on tende à cesser de discuter lorsque survient une situation de trafic difficile. Il n'en va pas de même lorsqu'on apprend tout juste à conduire, ce qui laisse à penser que l'interférence entre deux tâches dépend de la mesure dans laquelle elles ont déjà été apprises. Lorsqu'un stimulus donné est associé de manière répétée à une même réponse, il apparaît que les ressources attentionnelles, que cette liaison nécessite, diminuent progressivement, mais aussi qu'elles interfèrent de moins en moins avec les tâches concurrentes, acquérant ce que l'on appelle habituellement l'**automaticité**. C'est en poussant l'expert à être attentif lors de l'utilisation de ses processus de pensée automatique que nous l'aiderons à les mettre à plat. En effet, s'il prend garde de se protéger de ses automatismes acquis au cours de son apprentissage, en étant le plus attentif possible, alors des conflits devraient se produire en situation de tâches multiples. Ces conflits devraient permettrent à l'expert de faire émerger des incohérences dans ses connaissances et à être créatif pour se sortir de ce conflit.

- **Le contrôle de l'activité**

  Nous avons vu les problèmes que peuvent présenter le système attentionnel dans le processus d'élicitation de connaissances ou de création. Nous allons, maintenant, présenter un bon modèle du contrôle de l'activité, qui tient compte de la volonté du sujet au contraire des modèles précédents. Il nous sera ensuite possible de mieux modifier le système attentionnel de l'expert pour l'adapter au processus de création.

  - **Le système attentionnel superviseur**

    Le modèle du système attentionnel superviseur, proposé par Norman et Shallice, a pour but de fournir une vue générale du contrôle de l'activité en mémoire. Comme nous allons le voir, il fournit une base indispensable pour conceptualiser l'administrateur central de la mémoire de travail. Norman et Shallice étaient très intéressés par le fait de savoir comment les activités sont contrôlées et pourquoi ce contrôle échoue parfois. Ils expliquent que lorsque deux activités en cours entrent en conflit, il est alors nécessaire que la priorité soit donnée à l'une ou l'autre. Les décisions prises à ce niveau sont sous la dépendance d'un processus relativement automatique: le gestionnaire des priorités de déroulement, comprenant un ensemble de règles élémentaires, élaborées dans le système et qui peuvent

être déclenchées automatiquement. Le sujet qui agit est prisonnier de ses programmes habituels, en interaction avec tout ce que l'environnement présente. L'intervention de la volonté du sujet intervient alors par le biais du système attentionnel superviseur. La plupart des arguments présentés en faveur de ce modèle proviennent d'observations issues de la vie de tous les jours (actes manqués) ou de certains patients de neuropsychologie montrant une perte de contrôle du comportement.

- **Un exemple d'acte manqué**

  Un avocat quitte sa maison pour aller au travail le matin, préoccupé par quelque problème. Il entre dans son garage pour prendre sa voiture et s'aperçoit soudain qu'il a enfilé sa tenue de jardinage comme s'il allait travailler au jardin. On peut penser dans ce cas que le superviseur, ayant lancé le programme *aller au travail*, s'est ensuite préoccupé d'autre chose, laissant ainsi poursuivre le programme seul. De ce fait, la vision des bottes et habits de jardin, dans le contexte approprié, a suffit pour déclencher la routine jardinage, conduisant à l'activité inadéquate observée.

- **Synthèse**

  Ce modèle offre un début d'explication des mécanismes cognitifs à l'oeuvre dans le processus de conception. En effet, dans ce cas, le rôle du système attentionnel superviseur est de focaliser l'attention de l'expert sur certaines branches de raisonnement ou certaines formes de connaissances. Si on suppose que la capacité du système attentionnel superviseur est limitée, plus il y aura de points de vue en conflit, moins il sera capable d'éviter l'envahissement par des programmes stéréotypés. Cette intrusion de pensées stéréotypées peut alors conduire l'expert à des oublis ou des omissions, masquant ainsi des voies de recherche intéressantes. Il faut donc renforcer le système attentionnel superviseur, trouver une structure qui le soutienne, afin d'éviter qu'il soit surchargé et que des points de vue soient perdus.

8.1.1.2 Synthèse

Il semble donc qu'il existe un niveau supérieur où l'individu organise ses réflexions, ses indices perceptifs, en fonction de son intention, afin de leur donner la direction désirée. La conception est une forme de pensée où de nombreuses idées s'affrontent. Lorsque plusieurs objets se déplacent en même temps, il est difficile de les suivre tous. Le même phénomène semble également se produire dans tous les autres domaines de la pensée; plus les objets de notre pensée sont nombreux, plus il est difficile de concentrer notre attention sur l'un d'eux. Nous sommes obligés de nous en

Figure 8.1: L'assistant History de Comind garde une trace du cheminement de l'utilisateur. Il mémorise et affiche les assistants utilisés qui ont modifié l'état du problème. L'utilisateur peut ainsi revenir à un état où il était des heures auparavant.

tenir à un petit nombre d'entre eux et de perdre la trace des autres. Un miroir du cheminement des réflexions de l'expert semble un bon support pour conserver toutes les idées potentielles et ainsi éviter des actes manqués ou des processus de pensée stéréotypés. Ce miroir du cheminement semble, de plus, important pour augmenter l'attention de l'expert et l'impliquer davantage dans sa quête. De plus, la cognition d'un individu existe comme incarnée dans un univers extérieur, elle n'existe pas par elle-même. C'est donc en reflétant le cheminement des pensées de l'expert que nous lui permettront d'observer ses connaissances et ses stratégies de réflexion non pas de l'intérieur, comme une simple introspection, mais avec un regard plus aérien car incarné dans son milieu. L'utilisation d'un miroir reflétant l'activité de l'expert va donc permettre à celui-ci de développer une réflexion autonome sur son propre cheminement, de se désengager dans sa réflexion sur lui-même, par l'effet de distance entre le sujet réel et sa représentation.

8.1.2 L'assistant History

Puisque chaque personne possède une façon différente de concevoir, sur une longue période, les utilisateurs de notre système Comind devraient pouvoir observer leur propre cheminement cognitif à l'aide de l'assistant History. Ils peuvent choisir de suivre un ordre similaire ou de changer exprès leur interaction à l'aide de l'assistant History que nous allons présenter. Afin d'aider l'expert dans sa conception, il faut lui permettre, à la fois, d'ordonner sa connaissance et de la manipuler librement; c'est en lui suggérant des ≪voies de réflexion≫ oubliées, en le laissant libre de jouer avec toutes les analogies auxquelles il pense et en contrôlant qu'il ne s'écarte pas trop de l'objet de sa quête qu'il sera davantage créatif. En fait, le logiciel doit jouer le rôle d'un miroir déformant qui ne cesserait jamais de transformer la connaissance, en la présentant sous des angles différents, dans le but de remettre en question sa validité et de mettre en valeur des connexions jusque-là invisibles. En fait, cela doit être un outil capable de faire emprunter à l'expert des chemins de réflexion que, naturellement, il n'aurait pas parcouru. Mais, cela ne doit pas être simplement l'expert qui met de l'ordre dans sa connaissance, la machine doit suivre le même parcours et c'est ensemble qu'ils progresseront. Un miroir du cheminement des réflexions de l'expert est donc une bonne solution pour l'impliquer davantage

dans sa démarche créative; cela devrait permettre d'accentuer le caractère intentionnel de ses réflexions. Il pourra, ainsi, plus facilement focaliser son attention sur la voie de raisonnement qui l'intéresse. L'utilisation de l'assistant History, où chaque noeud représente une vue des connaissances de l'expert et de la machine à un instant donné, permettra d'illustrer ce miroir du cheminement. Comme tous les miroirs, il est infidèle. Néanmoins, dans l'amélioration du caractère intentionnel du cheminement de l'expert, la dynamique est plus intéressante que l'architecture physique de la mémoire. L'environnement de conception s'appuie sur un processus incrémental travaillant à partir d'une mémoire locale. Cette mémoire locale représente, l'état du problème connu par l'utilisateur et la machine. De nouveaux états peuvent être créés par l'expert et aussi par la machine afin d'approfondir ou de reformuler la conception. Au terme d'une session de travail avec l'environnement de Comind, un ensemble d'états est ainsi obtenu, qui représente les différentes étapes du processus de conception. Chaque utilisation d'un assistant dans le système Comind correspond à un état de connaissance momentané dans le raisonnement de l'expert. Ainsi, chaque utilisation d'un assistant modifie l'état du problème. L'utilisateur est libre de revenir sur un état donné et de l'approfondir dans une autre direction. L'assistant History, en plus d'être un bon outil pour naviguer dans son propre espace de recherche et l'agrandir, donne un sentiment de fermeture (voir section 8.2.2, page 89) de l'espace et rassure ainsi le concepteur.

8.2 L'assistant ShowCase

Les systèmes intelligents multimédia doivent non seulement aborder l'efficacité des algorithmes de recherche, mais également l'organisation de l'information. En outre, puisque les systèmes multimédia aident également les utilisateurs à résoudre des problèmes dans une série de question/réponse et d'activités de recherche/récupération, ces systèmes doivent être interactifs. L'intelligence de la machine et la multimodalité de l'information mettront en valeur les qualifications créatives des concepteurs. Dans cette section, nous décrivons nos expériences dans les systèmes multimédia pour supporter la conception. De tels systèmes contiennent des éléments sémantiques et multimodalement riches. L'organisation, la recherche et la présentation de ces éléments font partie des questions les plus fondamentales dans plusieurs disciplines: interaction avec l'information multimédia, interaction humain ordinateur, intelligence artificielle et recherche dans la conception [Pu and Lalanne, 1997].

8.2.1 Rechercher et interagir avec un système pour la conception

ShowCase est un tutoriel multimédia dont le but est d'enseigner à des étudiants la conception et les produits industriels. L'activité la plus importante auquel un utilisateur étudiant fait face, dans ce type d'environnement, est la recherche de l'information présente dans le système. La première page du système est montrée sur le schéma 8.2. Cette figure est basée sur le cours de méthodologie de conception de [Clavel, 1987]. Pour ceux qui ne connaissent pas ce cours, ce schéma suggère rapidement aux utilisateurs où ils peuvent trouver l'information qu'ils cherchent et deviner quel est le sujet qu'un élément illustre.

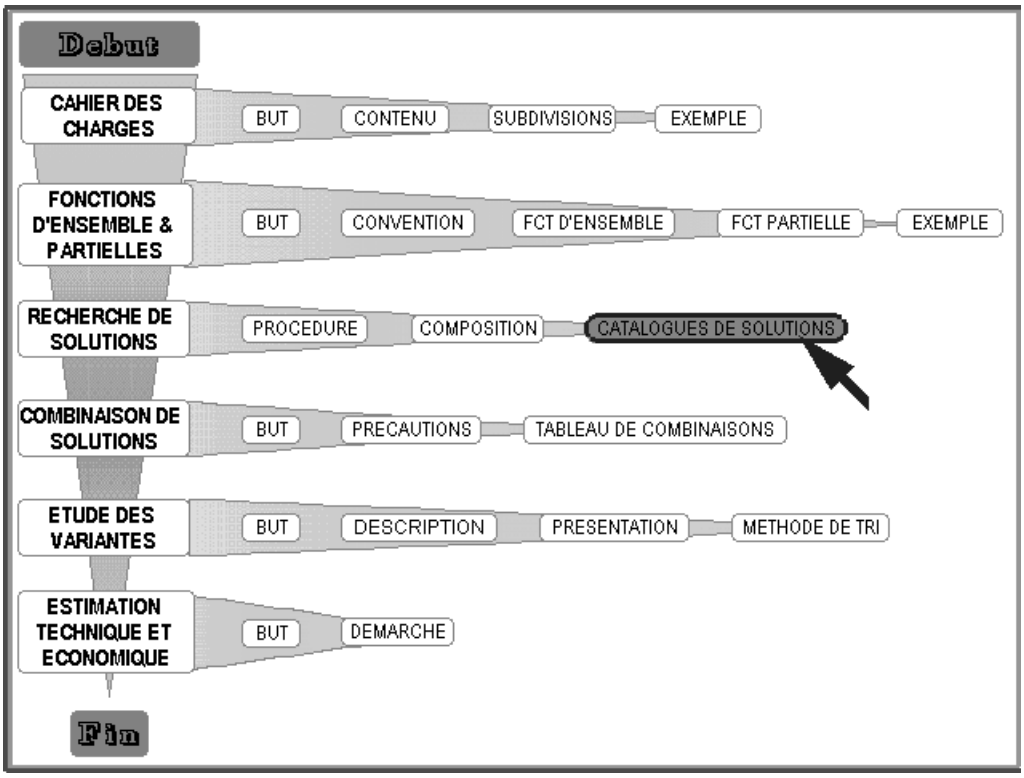

Figure 8.2: Le point d'entrée de ShowCase

8.2.1.1 Organisation

Afin d'établir un tel système, nous avons divisé l'information en deux couches: une couche sémantique et une couche syntaxique. Les éléments de base, qui peuvent être recherchés et parcourus, sont les entités dans la couche sémantique. Chaque élément représente une fonctionnalité particulière qui explique l'activité de conception. Les éléments qui résident à ce niveau sont organisés selon leurs objectifs didactiques plutôt que selon le type de modalité. Par exemple, la structure d'un

objet représente les relations topologiques et spatiales d'un produit. Cette structure est une description générale nécessaire pour chaque produit. Que la structure soit représentée par un croquis approximatif, une photo, ou un schéma dépend des documents disponibles et est donc décidé par la couche inférieure, la couche correspondant au type de modalité (la couche syntaxique). La couche syntaxique ou de modalité, est donc en dessous de la couche sémantique. Chacun des éléments de cette couche sert à illustrer une entité sémantique particulière. Souvent plusieurs éléments (des éléments multimodaux) de ce niveau supportent un même élément didactique de la couche supérieure. Un étudiant qui s'informe sur la conception peut donc rechercher de l'information telle que les premières étapes élémentaires du processus de conception, appelé le cahier des charges. Il peut choisir de continuer, horizontalement, le long de cette première voie d'accès et examiner ainsi chaque sous-module en détail. Notons qu'à ce moment, l'étudiant est seulement concerné par les fonctionnalités didactiques de chaque élément parcouru. Au niveau détaillé, des informations multimodales seront fournies pour cet élément.

8.2.2 Récupération et interaction avec l'information

Selon [Norman, 1988], toutes les interactions homme-machine ont lieu entre deux modèles, le modèle utilisateur du système et le modèle du concepteur du système. Les utilisateurs se renseignent seulement sur les comportements du système par l'intermédiaire du système image (celui qui apparaît aux yeux de l'utilisateur). Mais souvent l'image de système ne représente pas bien le modèle du concepteur, créant de nombreuses frustrations chez l'utilisateur, qui n'arrive pas à faire ce qu'il veut de la machine. Qu'un système de recherche et de récupération soit intelligent ou non, l'image du système (c'est-à-dire ce que l'utilisateurs peut rechercher et comment) joue un rôle clé.

Lorsqu'un système de documents multimédia est présenté à un groupe d'utilisateurs, chacun anticipe sur l'utilisation de ce système. De telles anticipations représentent l'espace des tâches cognitives des utilisateurs (ou le modèle utilisateur). Concevoir un système qui s'adapte bien à l'espace cognitif de chaque utilisateur serait impossible. Plutôt que de fournir un système intelligent qui se base sur leur espace de tâches, nous utilisons une stratégie de rencontre à mi-chemin (≪couper la poire en deux≫ pour les Vaudois). La conception du système se base sur la compréhension et la prévision d'un ensemble de comportements classiques de l'utilisateur. Cependant, le système est également établi avec une interface engageante et une quantité significative de transparence du système de sorte qu'il influence graduellement les comportements de l'utilisateur.

- **Le concept KISS** (Keep It Simple and Stimulating)

  Le système image de notre tutoriel est conservé simple au dépend de son intelligence parce que notre objectif est de faire sentir aux utilisateurs qu'ils sont en contrôle et qu'ils sont intelligents plutôt que de donner l'impression que le système est intelligent. Nous utilisons un modèle hypertexte pour la recherche. C'est une métaphore simple d'interaction qui suggère aux utilisateurs qu'ils peuvent soit parcourir le document, soit aller vers un autre document en cliquant sur un bouton, soit retourner au document précédent. Cette métaphore permet également à l'utilisateur d'accepter d'attendre afin de télécharger une animation coûteuse, parce que c'est lui-même, et non la machine, qui a commandé cette action en cliquant sur un bouton.

- **Sensation d'espace local et global**

  Les systèmes conventionnels d'hypertexte ne fournissent pas une représentation des relations globales et locales. Le World Wide Web est un exemple de cette imperfection. Par exemple, un étudiant dans notre laboratoire peut commencer à partir de sa Home Page à Lausanne et aller sur un site à l'institut Polytechnique de Worcester parce qu'il contient tous les laboratoires de recherches actifs impliqués dans la conception. Cette page Web peut alors le transporter vers de nombreuses universités d'Europe, d'Amérique du Nord, ou d'Asie. Après avoir suivi plusieurs pointeurs dans cet hypermonde, il a découvert une offre du travail à l'université de Maryland. Il est alors perdu dans son espace cognitif. C'est-à-dire, il a oublié pourquoi il était venu ici. C'est une expérience classique pour un explorateur de réseau (un ≪surfer≫ qui s'est laissé transporter par la vague du net) parce que les humains ont une mémoire attentionnelle limitée [Baddeley, 1990]. Quand nous sommes distraits par quelque chose qui n'était pas prévu et particulièrement quelque chose qui peut nous plonger dans un espace illimité d'exploration, nous perdons souvent notre objectif initial. Si le système de navigation pouvait indiquer à l'utilisateur où il se trouve, en relation avec ses étapes précédentes et lui indiquer où il désire aller, cette indication servirait de carte des activités mentales. Si, par exemple, le web browser offrait une mappe-monde et les points culminants où le navigateur se trouve et s'est trouvé, alors l'utilisateur aurait plus de contrôle sur le problème qu'il essaye de résoudre (comme par exemple chercher toute la littérature sur les méthodes de conception créatives).

  La première page de ShowCase fournit une vue générale de l'espace. Elle contient également un grand nombre de boutons qui changent de teinte lorsque

l'utilisateur se déplace dessus. Ainsi, il réalise immédiatement quels éléments sont cliquables et ceux qui ne le sont pas. Les boutons sont connectés à des documents qui contiennent plus de détails au sujet de la conception. Par exemple, le schéma 8.3 correspond à la vue locale d'une étape particulière de la conception appelée le catalogue de solutions. Ce catalogue établit un espace de solutions pour la conception. L'exemple montré représente les sous-fonctions du dispositif de verrouillage d'un bracelet de montre, à savoir les fonctions de blocage, de fixation et de guidage. Le bouton dans le coin supérieur-droit est une carte indiquant le rapport entre cette vue locale et la vue globale. L'élément en train d'être parcouru est mis en valeur dans cette vue iconisée de la première page.

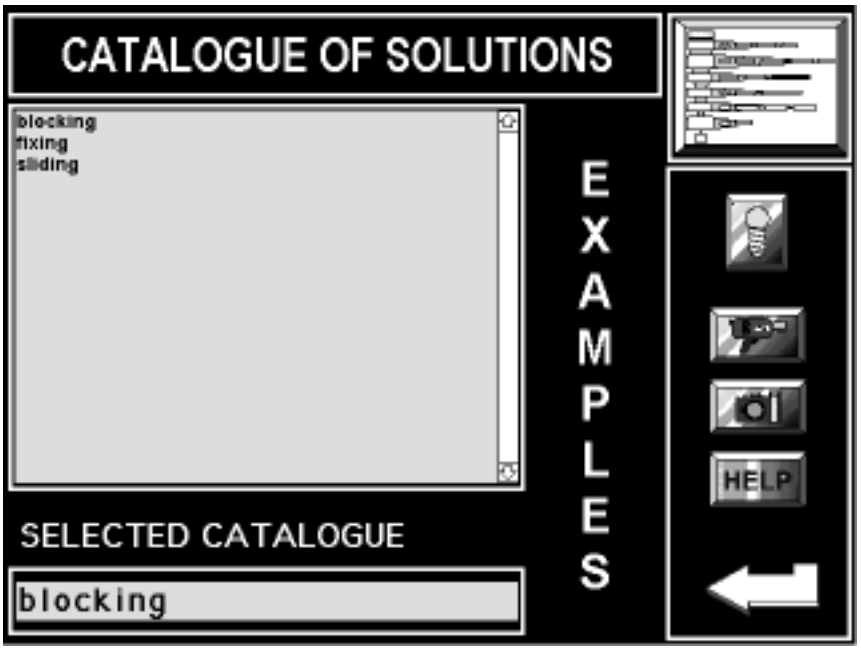

Figure 8.3: Trois fonctions d'un dispositif de verrouillage.

En plus d'être une carte générale, la première page, ainsi que sa vue iconisée, est une métaphore de l'espace cognitif de la tâche de l'utilisateur (celle qu'il va réaliser). Elle fournit à l'utilisateur un sentiment de fermeture. Le système est en un seul bloc et ne laisse aucune extrémité pendante. Selon [Laurel, 1986], un tel sentiment de fermeture, appelée ≪memisis≫, est un élément important dans les pièces de théâtre. C'est-à-dire, afin d'engager l'assistance dans un nouvel acte d'une pièce de théâtre et de les inciter à revenir après l'entracte, la fermeture du cheminement et de l'histoire doit être réalisé à la fin de chaque acte. Il est facile d'observer ce principe dans les longues séries télévisuelles américaines. La première fenêtre de ShowCase indique qu'afin de connaître la conception, tout ce qu'il faut savoir se trouve représenté dans cet espace.

C'est une information rassurante et encourageante pour les utilisateurs.

- **Consistance**

  Un monde d'hypertextes peut devenir facilement un labyrinthe si les éléments explorés ne se conforment pas à la compréhension de l'utilisateur du système. C'est-à-dire, peu après que l'utilisateur devienne familier avec le système, il établit un ensemble de règles. Si ces règles ne sont pas consistantes avec celles utilisées par le concepteur du système, l'utilisateur se sentira probablement vite perdu. Dans un tutoriel, des éléments qui servent le même but doivent permettre les mêmes modalités et fonctionnalités. Par exemple, les trois fonctions du système de fermeture du bracelet de montre (le blocage, la fixation et le guidage) possèdent la même palette de visualisation comme il est possible de le voir sur la figure 8.4. De plus, il y a deux espaces similaires de visualisation qui permettent aux étudiants de voir des exemples de conception de tels mécanismes dans de multiples modalités. Un étudiant peut par exemple visualiser une animation, un croquis, ou les deux en même temps.

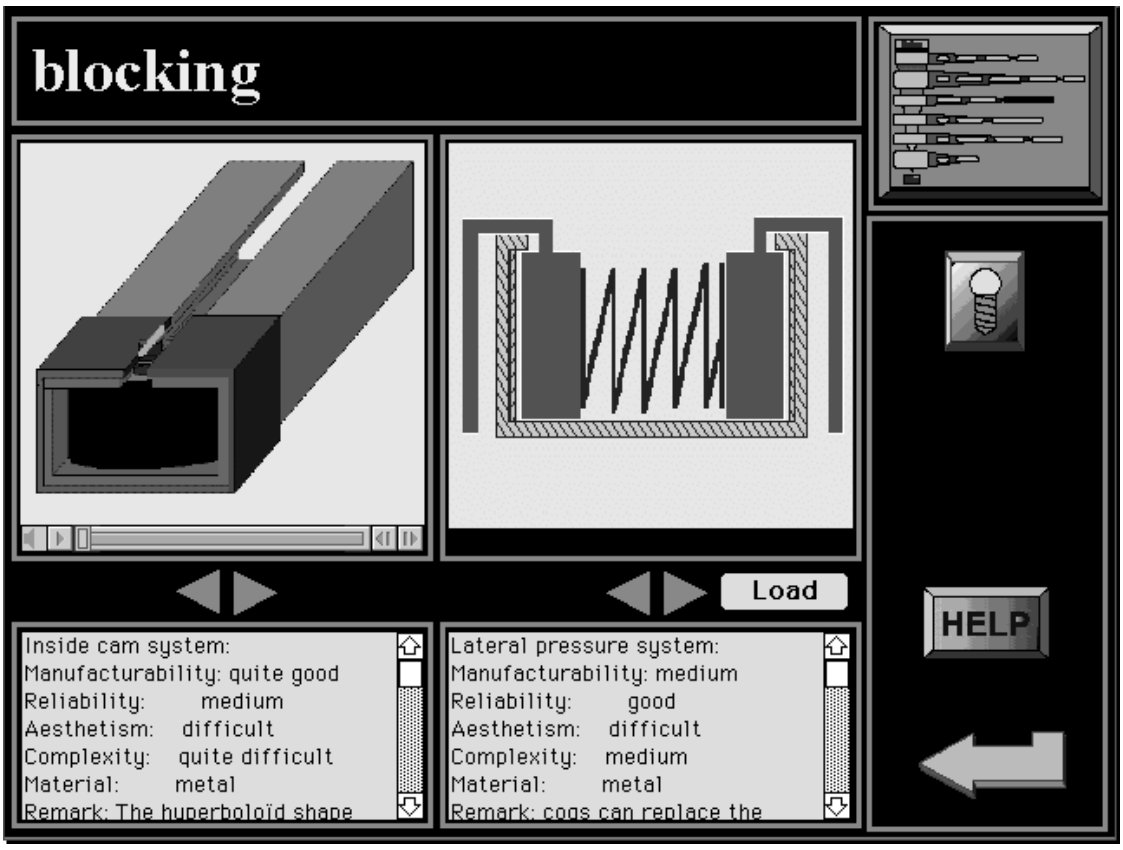

Figure 8.4: La fonction de blocage.

8.2.3 Recherche de cas multimedia

CORINTH [Pepin, 1893] est un système de raisonnement à base de cas qui supporte la conception en se fondant sur des expériences dans divers domaines: la conception à base de contraintes, la conception de ponts et la conception ≪conceptuelle≫.

8.2.4 Evaluation

Nous n'avons fait aucune évaluation expérimentale scientifiquement correcte afin d'évaluer notre systèmes ShowCase. Cependant de nombreux étudiants et des professeurs du domaine de la conception ont été invités à utiliser le système ShowCase. Nous avons été récompensés par cette utilisation parce que le système ne tombe jamais en panne. Selon notre analyse, le fait que le système ne tombe pas en panne est une réflexion de la bonne coopération entre les utilisateurs et la machine. Le mouvement adéquat est évident pour les utilisateurs de sorte qu'ils ne ≪tournent pas autour du pot≫ (une cause généralement importante dans les pannes d'un système). Le deuxième crédit est qu'ils ont trouvé le système facile et amusant à utiliser dans une courte période de temps. Nous pensons donc que la vue globale a correctement reflété ce qui est contenu dans le système de sorte que les utilisateurs ne se sont pas perdus dans l'espace de recherche. En retour, ils ont obtenu un sentiment de fermeture de leur tâche cognitive. Aussi, quoique nous n'ayons pas fait une étude expérimentale de notre système, nous avons appliqué les méthodes existantes dans l'interaction humain-ordinateur, qui ont fait leurs preuves et on été éaluées par d'autres chercheurs.

8.2.5 Synthèse

La conception implique de nombreuses formes différentes d'information et de médias. Leur intégration dans un système d'aide à la conception est une tâche ardue. Les concepteurs d'un tel système doivent considérer soigneusement l'organisation des éléments multimodaux. Ils doivent également fournir une étude interactive de l'environnement aux utilisateurs de sorte qu'ils puissent établir un modèle conceptuel du comportement du système. L'interaction dans la recherche de documents multimédia devient ainsi une question de l'interaction humain-ordinateur. Dans cette section, nous avons défini et discuté ces questions dans le contexte du système ShowCase.

Chapitre 9

Configuration d'un lotissement

9.1 Définir le problème

Nous considérons maintenant un problème de conception inspiré du travail de Navinchandra [Navinchandra, 1991] afin d'illustrer l'utilisation de notre système. Une région au sud-ouest d'une ville doit être développée. Huit lots de terrain sont disponibles pour de futurs développements. Six différentes sections sont à développer: une aire de récréation, un immeuble, un lotissement de 50 villas, une grande décharge, un cimetière et une école. Nous considérons ici que les caractéristiques du site ont été déterminées en utilisant un système standard pour saisir des plans. Un plan du site est représenté dans ShowCase, un des assistants de COMIND permettant de stocker des documents multimédias de conception (figure 9.1). Le plan présente huit terrains disponibles pour un futur développement. Les lots 3, 5, 7 et 9 sont des sites relativement plats avec un sol relativement stable pour une construction. Les lots 10 et 12 sont un peu en pente dans une jolie région boisée. Le problème consiste à trouver la meilleure assignation de lots aux différents types de développement. Une conception est complète lorsqu'à chaque type de développement (immeuble, maisons,...) est assigné un terrain. La conception finale doit satisfaire l'ensemble des contraintes suivantes:

1. la décharge publique ne doit ni être visible depuis les maisons, ni depuis l'immeuble.

2. les terrains en pente doivent être évités pour les constructions qui nécessitent des fondations.

3. les sols instables doivent être évités pour les constructions qui nécessitent des fondations.

Afin de définir le problème, le concepteur utilise tout d'abord l'assistant Brainstorming. A cette étape, le concepteur peut taper librement ses idées à propos du

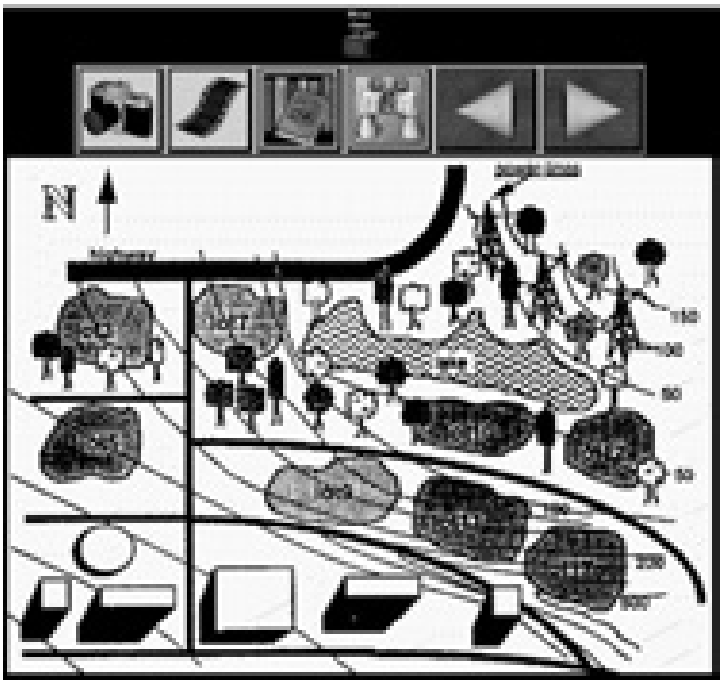

Figure 9.1: Le plan du lotissement présenté dans ShowCase.

problème dans un éditeur.

Après la production d'un bon nombre d'idées, le concepteur peut utiliser l'assistant Param-Def afin de définir le problème de façon plus formelle et être capable de le résoudre. Notons que l'utilisation de l'assistant Brainstorming n'a rien d'obligatoire. Chaque étape peut être plus ou moins complétée, plus ou moins souvent parcourue, au libre choix de l'utilisateur. Le concepteur fixe six paramètres ayant respectivement pour nom: recre, appartment, houses, dumpsite, cemetery et school. Chaque paramètre possède le même domaine de valeurs: {3, 5, 7, 9, 10, 11, 12, 17} correspondant aux huit terrains disponibles. Le concepteur définit ensuite les trois contraintes du problème à l'aide de règles, qu'il aurait aussi bien pu définir à l'aide de matrices (figure 9.3).

9.2 Résoudre le problème

Dans notre scénario, le concepteur utilise ensuite l'assistant Solve afin de trouver des solutions à son problème. Au cours de la recherche, 12% de l'espace a été parcouru et déjà 104 solutions ont été trouvées. La visualisation de l'espace exploré montre que l'algorithme passe pratiquement par toutes les solutions potentielles possibles. Parce que l'algorithme est rarement stoppé dans ce cas, le concepteur décide d'arrêter la course de l'algorithme et de revenir à l'assistant Param-Def pour y ajouter des

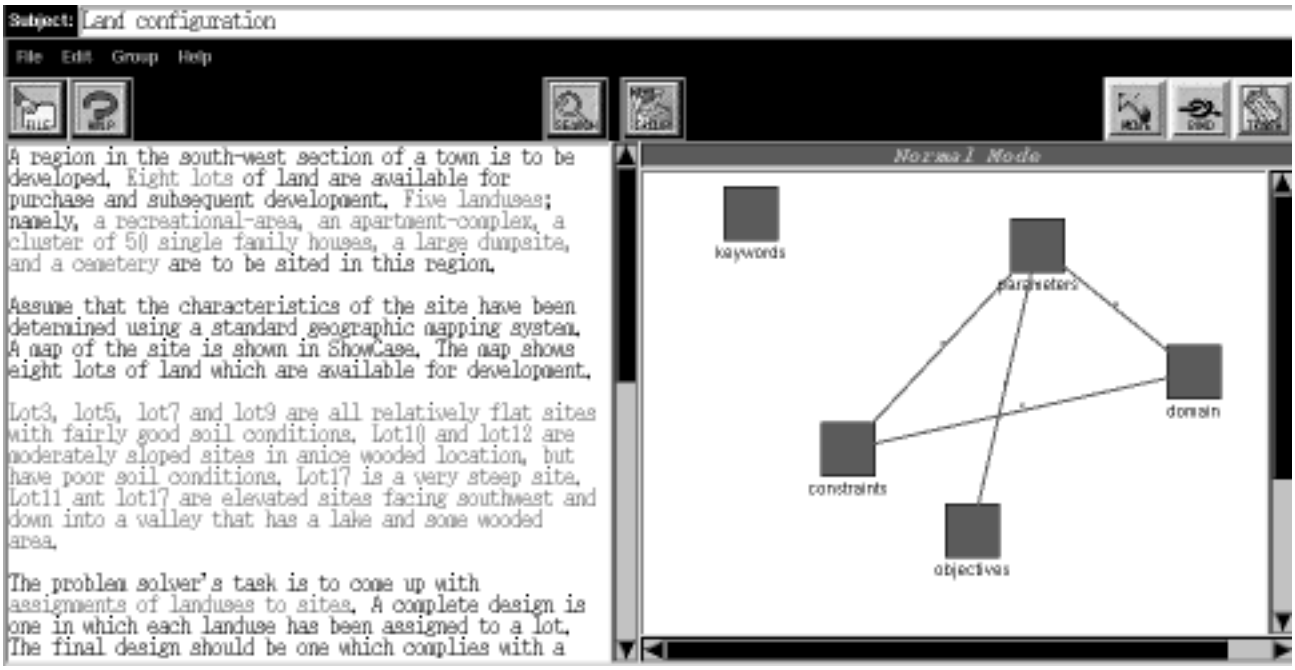

Figure 9.2: L'assistant Brainstorming de Comind représentant le problème. A gauche, un editeur permet à l'utilisateur d'exprimer librement ses idées. A sa droite, il peut regrouper ses idées dans des groupes, qu'il peut organiser entre eux afin de faire emmerger une structure.

contraintes:

- **4.** l'aire de récréation doit se trouver près de l'eau.
- **5.** l'autoroute est bruyante et laide et donc les appartements, les maisons et l'aire de récréation ne doivent pas la côtoyer.

Le concepteur retourne ensuite vers l'assistant Solve. Cette fois, la visualisation de l'espace de recherche indique que bon nombre de solutions sont évitées. Le parcours de l'espace se fait donc assez rapidement. Finalement, après quelques secondes, 24 solutions sont trouvées (figure 9.3). C'est un nombre encore trop important pour pouvoir les comparer à l'oeil nu. Le concepteur appelle donc l'assistant TradeOff.

9.3 Trouver la bonne solution parmi plusieurs

L'assistant TradeOff fournit un éditeur pour définir les critères de la conception. Dans notre scénario, le concepteur définit deux objectifs afin de faire émerger les meilleures solutions: le coût total et le total de la nuisance sonore. Le prix des lots variant suivant la qualité du sol, la situation et aussi suivant le type de construction que le concepteur veut placer dessus. Quant à la nuisance sonore, il est bien évident

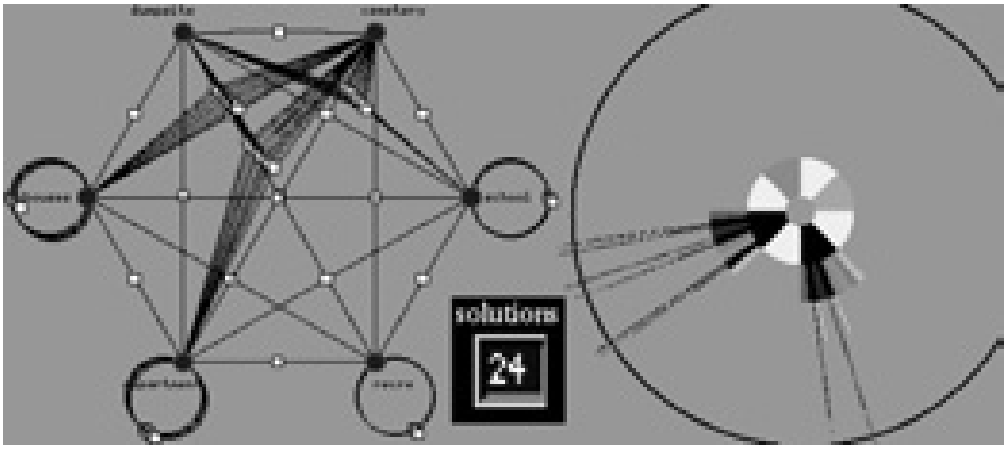

Figure 9.3: La définition du problème à l'aide de l'assistant Param-Def de COMIND et sa résolution à l'aide de l'assistant Solve de COMIND.

que la tolérance au bruit n'est pas la même pour une maison que pour une décharge. Ainsi il est par exemple plus convenable de placer la décharge près de l'autoroute que les maisons. Plusieurs visualisations sont proposées au concepteur afin d'évaluer les solutions selon les critères définis dans l'éditeur. Le concepteur peut aussi évaluer la pertinence du choix de ses critères.

Plusieurs scénarios sont imaginables à cette étape. Le concepteur pourrait, par exemple, se rendre compte qu'aucune des solutions trouvées ne lui convient parce que celles qui l'intéressent ont été évitées. Le concepteur devrait alors réviser la définition de ses contraintes en utilisant les outils adéquats pour trouver quelles sont les contraintes qui lui ont fait éviter les solutions qu'il désirait. Nous avons choisi un autre scénario parmi la multitude possible. Dans notre cas le concepteur n'est pas satisfait par les solutions optimales selon ses critères. Les configurations les moins chères et les plus tolérables en terme de nuisance sonore ne sont pas de bonnes solutions selon lui. Les maisons sont loin du lac, le cimetière est placé dans une région plutôt agréable , etc. Le concepteur réalise donc que la vue est un critère important dans son évaluation et retourne vers l'éditeur de l'assistant TradeOff afin de définir ce nouveau critère. Remarquons ici l'importance que joue la visualisation pour la découverte de critères pertinents si l'on néglige la simplicité de l'exemple. Pour d'autres raisons le concepteur définit aussi un autre critère: la distance entre l'école et les habitations. La figure 9.4 présente différentes visualisations choisies par l'utilisateur afin de comparer les solutions.

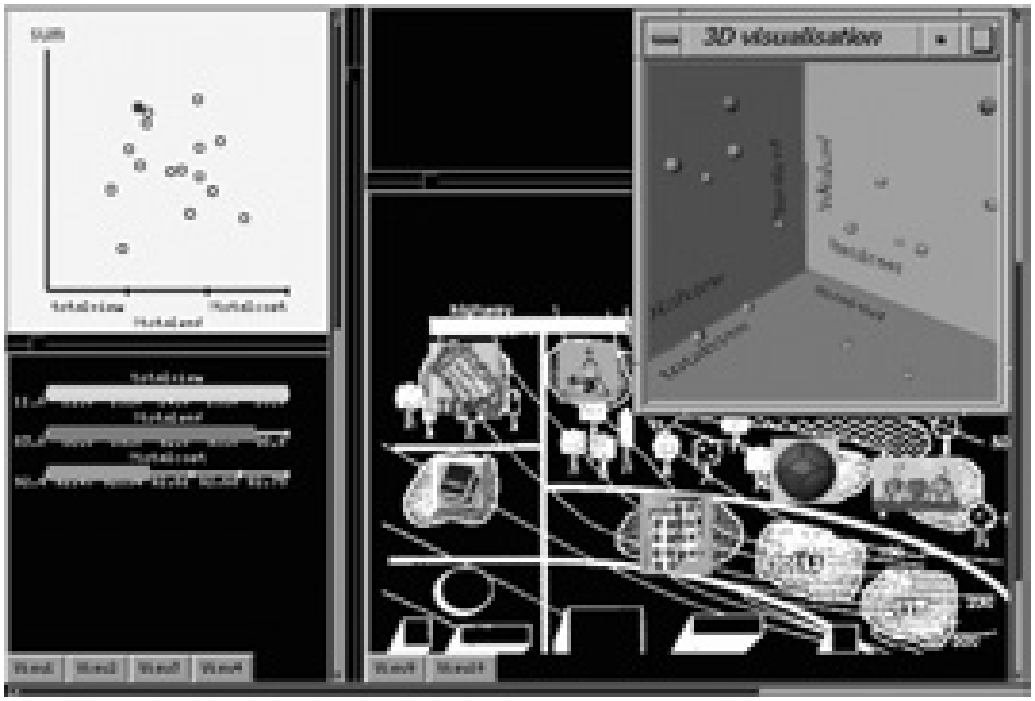

Figure 9.4: Comparer les solutions grâce à l'assistant TradeOff.

9.4 Relaxer le problème

Notre scénario continue, le concepteur veut maintenant ajouter un supermarché dans son lotissement. L'assistant Param-Def est à nouveau appelé et le concepteur ajoute à sa définition le paramètre supermarket. Il définit, principalement pour des raisons esthétiques, que le supermarché ne peut pas se trouver en face des maisons, de la décharge et du cimetière. L'assistant Solve est ensuite appelé. Aucune solution n'est trouvée. Le problème est sur-contraint. Le concepteur doit donc trouver les conflits implicites dans la définition du problème. Parce que le problème est assez grand et parce qu'il est difficile pour un humain de raisonner sur plusieurs contraintes et paramètres à la fois, l'assistant Conflict Elicit propose à l'utilisateur la distribution topographique des ensembles de contraintes violant l'ensemble des combinaisons possibles. Lorsque le nombre de paramètres et la taille de leur domaine augmente, la première famille d'algorithmes (voir section 7.3.1, page 63) nécessite une calculation trop coûteuse. Pour cette raison, nous utilisons notre algorithme d'élagation (voir section 7.3.2, page 71). Nous vous invitons à aller à la section 7.4.3, page 78, pour voir comment le problème peut être par exemple relaxé dans ce scénario. D'autres relaxations auraient pu être imaginées. A l'aide de l'algorithme d'élagation basé sur la méthode de Knuth, le concepteur réalise que les règles concernant le supermarché et les maisons, le supermarché et les appartements, ainsi que les maisons et les appartements, sont les trois couples de paramètres les plus bloquants. Parce que

le supermarché ne peut pas faire face à la majorité des constructions et parce que la décharge et le cimetière restreignent déjà la taille de l'espace de solutions, de nombreuses solutions sont bloquées.

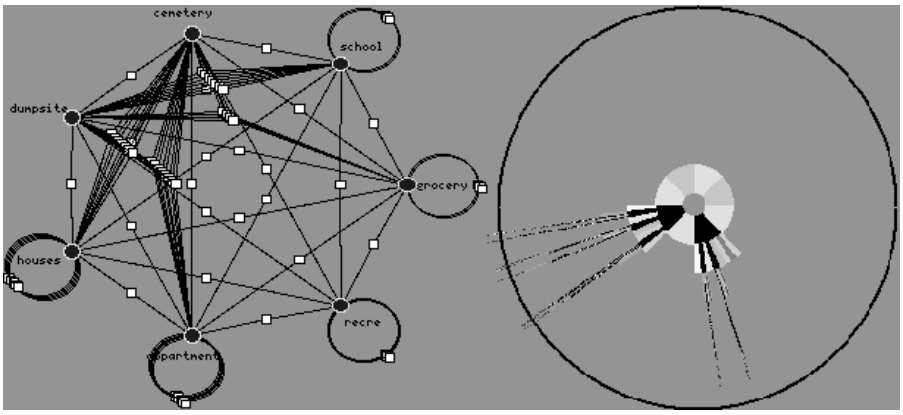

Figure 9.5: La définition du problème et sa résolution. Le supermarché a été remplacé par une épicerie.

L'utilisateur décide donc de remplacer le supermarché par une épicerie plus discrète et s'adaptant mieux au paysage. Finalement 16 solutions sont trouvées (figure 9.5).

9.5 Réflexion du processus de conception

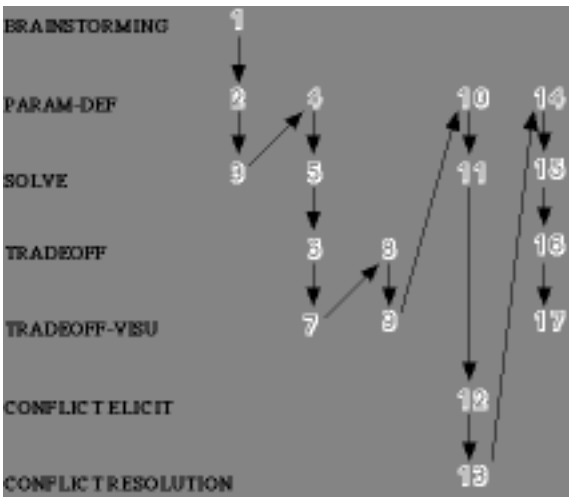

Figure 9.6: L'historique du scénario d'utilisation

La figure 9.6 illustre l'historique du scénario présenté dans ce chapitre. L'assistant Hystory garde cette trace. Une réelle conception possède davantage d'étapes. Nous avons présenté les plus représentatives dans ce chapitre.

Chapitre 10

Structures parallèles de robot

Nous avons vu qu'il est possible d'intégrer dans Comind une interface supplémentaire dédiée à un type de problème (voir section 4.4, page 39). Nous montrons dans ce chapitre un exemple basé sur la génération de structures parallèles robotiques.

10.1 Définir un robot

Notre système a été d'une grande utilité pour explorer les connaissances du domaine et les reformuler et également comme outil de réflexion. Pour plus d'informations sur la conception de mécanismes, voir [Rolland, 1998]. L'utilisateur, en plus d'interagir avec l'éditeur dédié à la définition de robots, peut aussi directement intervenir dans l'assistant Param-Def lorsqu'il le maîtrise suffisamment. La figure 10.1 montre l'éditeur pour les robots ainsi que les contraintes définies à l'aide des glisseurs (sliders) et leur traduction dans l'assistant Param-Def. La traduction est dynamique, c'est-à-dire qu'une modification dans l'éditeur de robots modifie automatiquement, et instantanément, l'assistant Param-Def. Ils partagent la même connaissance du problème, à tout moment, comme tous les assistants de Comind.

10.2 Deux étapes de recherche

Il y a deux grandes étapes de recherche possibles dans l'éditeur de robot. La raison pour laquelle nous avons divisé la recherche en deux est principalement liée au temps de calcul. Les sliders à gauche permettent de trouver des structures. Chaque ascenseur permet de définir une contrainte différente. Pour le premier type de recherche, les sliders sont:

- le nombre de boucles que possède la structure
- le nombre de joints

- le nombre de bras sur la base
- le nombre de bras sur la plateforme mobile
- la profondeur du système, c'est-à-dire le nombre de bras maximum entre la base et la nacelle.
- la largeur maximum d'un sous-système, c'est-à-dire le nombre de bras maximum qui partent d'un embranchement, excepté la base.

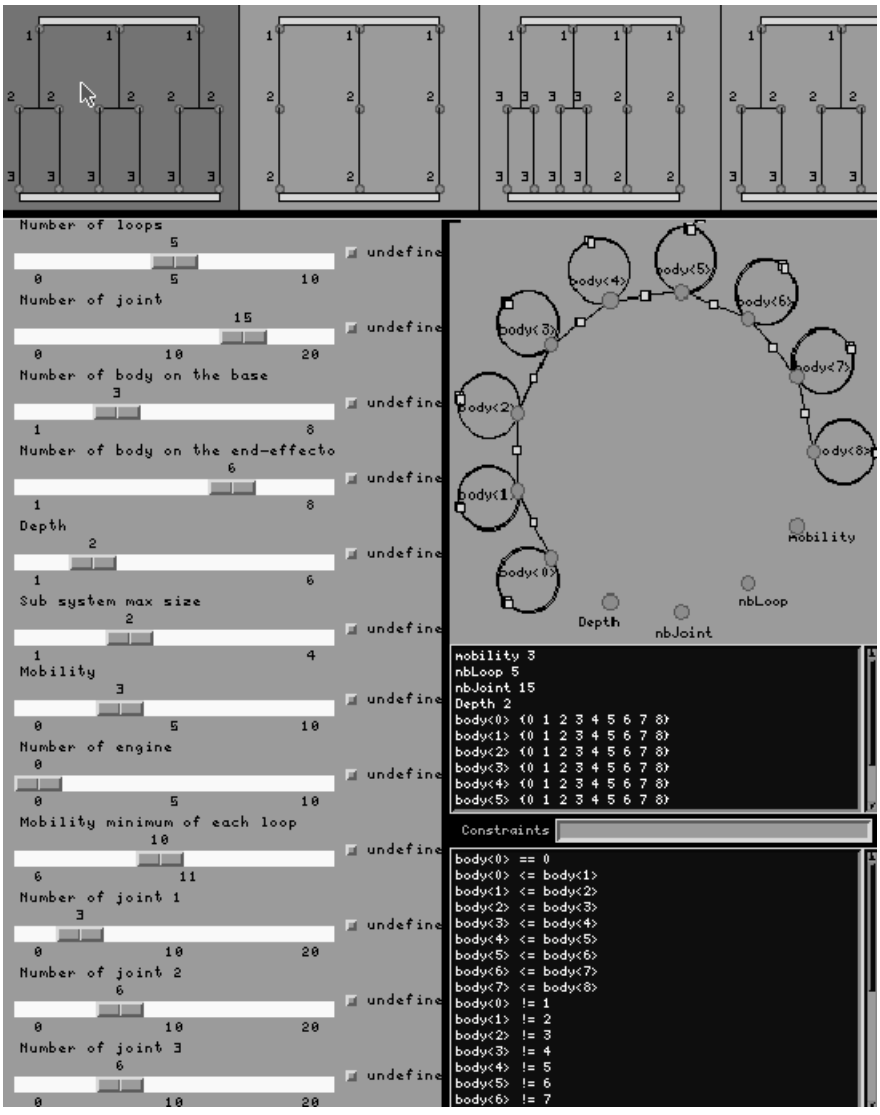

Figure 10.1: L'assistant RobotDef, un méta *Param-Def.* Les contraintes sont définies à l'aide des sliders dans l'assistant RobotDef et automatiquement traduites dans l'assistant *Param-Def.* Les autres assistants de Comind sont alors disponibles et peuvent coopérer avec l'assistant Robot-Def.

Comme il est possible de le constater, ces contraintes sur-définissent le problème. Parfois, elles peuvent être redondantes. Cependant, en général, elles permettent à l'utilisateur de focaliser sa recherche.

Pour le second type de recherche, qui permet de trouver le type de joint et de calculer la mobilité, d'autres contraintes peuvent être définies:

- la mobilité
- le nombre de moteurs
- la mobilité minimum pour une boucle (minimum 6, en éliminant le cas plan)
- le nombre de joints à un degré de liberté (rotatif, prismatique, hélicoïdal)
- le nombre de joints à deux degrés de liberté (cardan, etc.)
- le nombre de joints à trois degrés de liberté (rotule)

Chaque contrainte peut être définie ou non suivant que la recherche est désirée précise ou non (facteur de vitesse aussi). Par exemple, l'utilisateur peut choisir de ne pas définir la mobilité. La mobilité deviendra alors un critère de comparaison. L'utilisateur peut aussi utiliser une structure de robot existante et modifier une ou plusieurs de ses contraintes pour en trouver une variante. Il peut aussi dessiner une structure, grâce à l'éditeur de dessin de robot, en utilisant seulement la souris. Ensuite. il peut chercher les valeurs des joints pour la structure ou seulement pour une partie donnée et spécifier les contraintes qu'il désire (mobilité, moteurs, etc). L'utilisateur peut aussi obtenir une représentation en trois dimensions de la structure

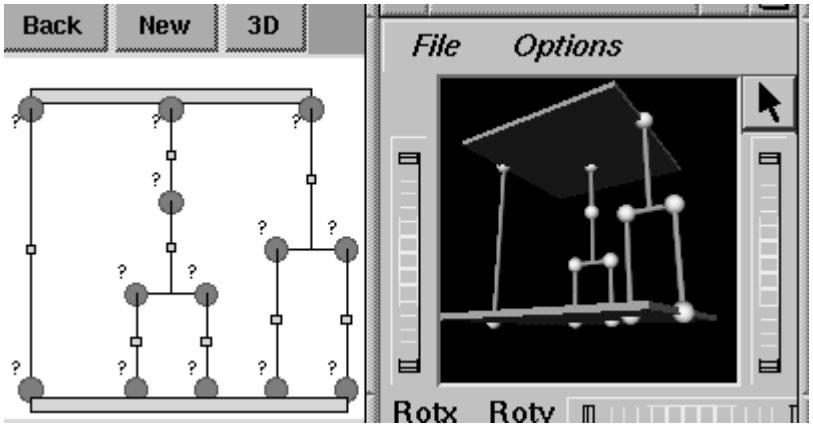

Figure 10.2: Le robot dessiné dans l'éditeur est automatiquement représenté en trois dimensions sur demande de l'utilisateur.

qu'il a dessinée en deux dimensions comme le montre la figure 10.2. Cela peut fournir une idée de la forme de la structure.

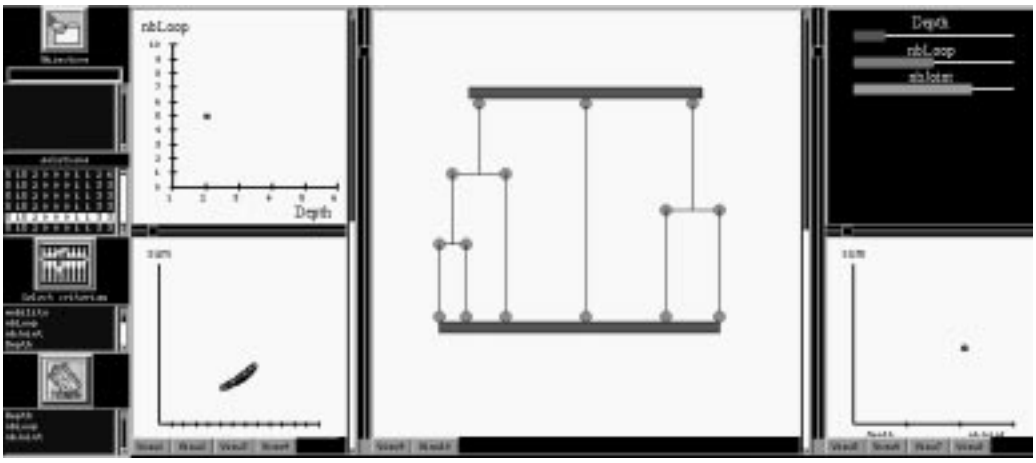

Figure 10.3: Pour une petite structure, des solutions sont rapidement trouvées

10.3 Résoudre, comparer, relaxer, etc

Puisque l'éditeur de robot communique avec l'assistant Param-Def, tous les assistants de Comind sont disponibles. Par exemple, la figure 10.3 montre le résultat d'une petite recherche. L'utilisateur peut alors utiliser l'assistant tradeOff pour comparer ses solutions. Nous avons défini un problème simple pour mieux illustrer notre

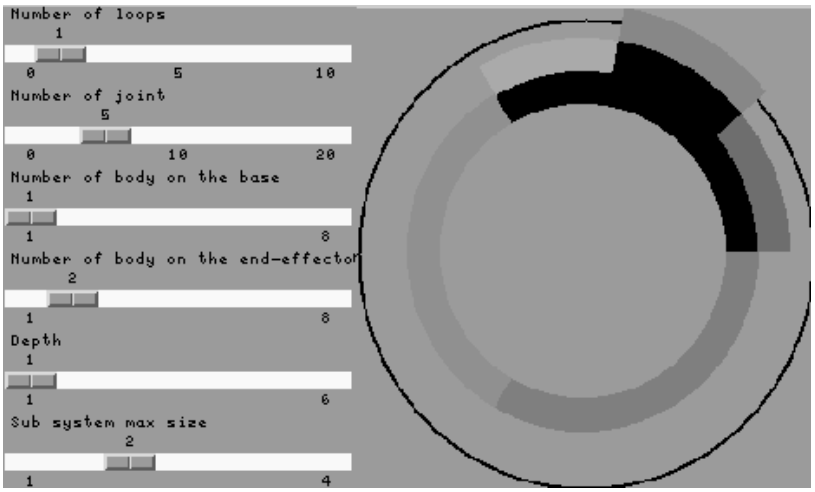

Figure 10.4: La définition du robot et sa résolution, pas de solution.

démarche. La figure 10.4 représente la définition du problème. Nous recherchons un robot possédant:

- une boucle
- 5 joints
- 1 joint sur la base

- 2 joints sur la nacelle
- de profondeur 1
- possédant un sous-système de taille maximale 2.

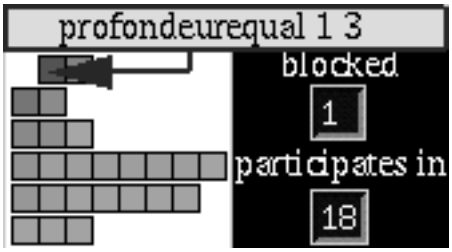

Figure 10.5: La profondeur est responsable.

Nous avons alors fait une recherche. Aucune solution n'est trouvée. La visualisation de l'assistant Solve donne déjà le coupable (figure 10.4). C'est la profondeur qui est trop petite. Cependant, pour plus de clarté, nous avons utilisé l'assistant Conflict. La visualisation de la première famille d'algorithme sous forme de lattices (figure 10.5) nous indique l'ensemble de contraintes le plus petit et le plus responsable de l'échec de la recherche (plus le carré, représentant un ensemble de contraintes, est foncé, plus il bloque). Il limite la profondeur à 1. La visualisation nous indique aussi que cette contrainte bloque 1 solution possible. La visualisation nous apprend

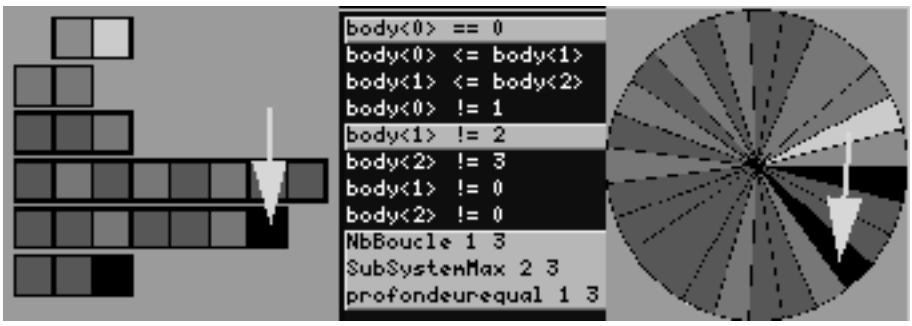

Figure 10.6: Un conflit

aussi qu'il y a un conflit plus petit que le problème lui-même. Le conflit apparaît tout en noir dans le lattice du bas (figure 10.5). Nous l'avons sélectionné, dans la visualisation camembert (figure 10.6, identique en information à la précédente). Ce conflit exprime simplement que l'on ne peut pas interdire en même temps:

- au deuxième bras de ne pas être sur la base
- au nombre de boucle de valoir 1

- d'avoir un sous-système de largeur maximum 2
- et d'avoir une profondeur valant 1.

Nous décidons de façon arbitraire, d'autres relaxations étant possibles suivant les objectifs, de mettre la profondeur à 2 (figure 10.7). Et cette fois, en relançant la recherche, nous obtenons une solution (figure 10.8).

Figure 10.7: Modification du problème initial

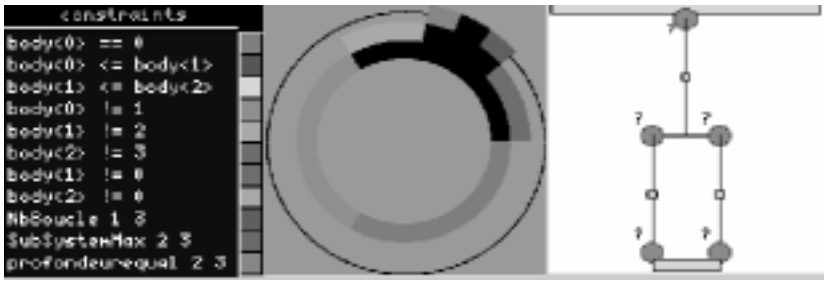

Figure 10.8: Une solution est obtenue

10.4 Remarques

Cette application peut être un outil intéressant de réflexion. Cependant, ce n'est qu'un exemple permettant de montrer qu'il est possible de concevoir un outil plus intuitif que l'assistant Param-Def afin de définir un problème particulier. Afin de mieux soutenir la conception de robots, il reste des efforts à fournir pour rendre l'outil vraiment intéressant. Toutes les connaissances n'ont pas été formalisées. De plus, le lien avec un autre outil d'aide à la simulation, interactive et temps réel, de structures robotiques [Fluckiger et al., 1996] reste à faire. Il sera alors possible de générer des structures suivant un ensemble de contraintes (mobilité, etc) et de comparer ces solutions grâce à des critères et même grâce à la simulation. Il nous semble qu'il serait particulièrement intéressant de continuer ce travail.

Chapitre 11

Synthèse et contributions

11.1 Deux points

La direction initiale de la thèse était comment l'homme et la machine peuvent former une synergie de leur intelligence dans un environnement de résolution de problèmes. Nous avons trouvé un modèle d'intelligence interactive basé sur le concept de la connaissance et du calcul distribués, un modèle d'interaction entre les concepteurs humains et un ensemble d'assistants logiciels. La thèse a été distribuée suivant deux grandes questions. La seconde question a aidé à étudier la première d'un point de vue global:

1. comment l'homme et la machine peuvent-ils former une synergie de leur intelligence dans un environnement de résolution de problèmes? Notre modèle d'intelligence interactive, basé sur la complémentarité des intelligences, contribue-t-il à cette synergie?

2. ce modèle s'adapte-t-il à la conception créative? Dans cette thèse, nous avons proposé un modèle d'interaction pour la conception créative basé sur une architecture de résolution des problèmes distribuée entre l'utilisateur et un ensemble d'agents. Nous avons montré qu'un tel environnement n'interdit pas aux concepteurs humains d'être créatifs, d'augmenter leurs qualifications pour des tâches de conception; il pourra les aider à définir, à explorer et à évoluer dans l'espace de conception à l'aide des techniques de visualisation.

11.2 Contribution

L'objectif principal de ce travail était de montrer qu'une collaboration humain-machine pouvait augmenter les facultés de l'humain dans une tâche de résolution de problèmes. Nous avons choisi la conception créative comme exemple particulièrement explicite du monde ouvert dans lequel nous nous trouvons. Nous avons

soutenu, en particulier, deux processus cognitifs d'une conception créative supportés respectivement par l'assistant TradeOff et l'assistant Conflict. De nombreux scénarios dans différents domaines de conception et la participation de plusieurs utilisateurs, nous ont permis de constater que l'humain pouvait, à l'aide de notre système, résoudre des problèmes sous et sur-contraints, problèmes que nous pensons être à la base des processus cognitifs de création. Nous montrons dans les sections 11.2.1 et 11.2.2 les résultats que nous avons obtenus dans chacune des tâches où une collaboration humain-machine s'avère particulièrement efficace.

11.2.1 Conception créative assistée par ordinateur: un exemple d'intelligence interactive qui fonctionne

La deuxième question a abouti à la réalisation d'un environnement supportant la conception créative. La conception est un exemple dans lequel la synergie entre l'homme et la machine est particulièrement utile en raison de la complexité croissante du monde (critères du consommateur de plus en plus spécifiques et période décroissante du cycle de vie d'un produit). Les créateurs font souvent face à des problèmes énormes contenant beaucoup de paramètres, des contraintes et des critères. L'intelligence artificielle offre de nombreuses techniques pour la résolution de problèmes. Un bon modèle d'interaction entre l'intelligence de l'homme et les compétences de la machine peut rendre disponible ces techniques aux créateurs et leur fournir un outil supportant la mémoire, la perception et le calcul. De cette façon, les créateurs peuvent se concentrer davantage sur leur propre créativité alors que la machine prend soin des tâches de routine. Nous avons présenté dans cette thèse ce modèle et son application dans les différentes tâches que nous avons trouvé importantes pour aider la conception créative. Nous avons choisi de supporter principalement la définition du problème, sa solution, son optimisation et sa relaxation dans le cas d'une définition de problème sur-contraint. Cette distribution de tâches suit les différents processus cognitifs auquel un créateur peut avoir à faire face. Nous avons validé notre modèle en comparant le modèle de concepteur du système au modèle utilisateur, afin de créer une image correcte de système. Souvent, le modèle mental du système est différent selon que l'on est du côté du concepteur ou du côté de l'utilisateur. Ce concept était particulièrement important dans notre problème parce que la synergie entre l'utilisateur et le système dépend de l'exactitude du modèle de l'utilisateur du système (voir 8.2.2, page 89). Différents exemples de conception ont été traités pour valider notre architecture. Puisque l'environnement de cette thèse est un département de microtechnique, nous avons expérimenté notre système dans quelques domaines scientifiques du département. Un exemple complet, la conception de structures parallèles de robot, une des spécialités du département dont nous faisons partie, a été réalisé afin d'aider un assistant dans son travail de

conception et de réflexion.

- Nous avons collaboré avec des concepteurs de structures parallèles de robot. Notre système a été d'une grande utilité pour extraire les connaissances dans le domaine [Rolland, 1998] et aussi comme outil de réflexion. Il reste cependant des efforts à fournir pour rendre l'outil vraiment intéressant.

- L'architecture de notre système est d'ores et déjà utilisée. Le projet MicroCE a pour principal objectif l'aide à la conception de produits microtechniques. Ce projet est le fruit d'une collaboration interne (ISR-LICP) et externe (MECANEX). Il met dans la même boucle un système d'aide à la conception créative, inspiré de notre système, et un système d'aide à la fabricabition et à l'assemblage.

- Nous avons construit un prototype mettant en avant le principe d'intelligence interactive, capable de soutenir la conception créative. Il valide nos pré conditions (distribution autour de l'utilisateur, connaissance partagée, respect de la complémentarité, visualisation comme espace interactif) afin d'aider un utilisateur au niveau de son raisonnement. Ce système, Comind, est relativement stable et complet. Il couvre le champ de la conception et a été appliqué à de nombreux domaines. Il peut ainsi servir de modèle pour construire d'autres applications particulières d'aide à la conception.

- Cette thèse fournit donc un exemple fonctionnant du concept d'≪intelligence interactive≫. Le modèle peut être appliqué à n'importe quel domaine traité par l'intelligence artificielle et il procure de nouveaux outils visuels et de calcul afin d'**aider** les humains à concevoir, diagnostiquer, projeter et prendre des décisions.

- L'audience qui bénéficiera le plus de notre système est la communauté des concepteurs. Cette communauté est très large. Nous ajoutons donc que notre système intéressera surtout les concepteurs qui souhaitent avoir un outil de réflexion et d'exploration. Notre approche se caractérise en ce qu'elle prend au maximum en compte les différences entre les facultés de l'humain et celle de la machine afin de mieux les faire collaborer.

11.2.2 De nouveaux outils de visualisation

La deuxième approche (application de l'intelligence interactive à la conception créative) a permis de prouver que notre modèle d'intelligence interactive n'interdit pas la créativité et améliore les qualifications de l'humain pour définir, explorer et évoluer dans l'espace de conception. Elle prouve la validité du modèle pour une résolution

de problèmes prise dans sa globalité. Elle a ainsi validé la valeur de l'architecture ouverte.

Néanmoins, afin de traiter complètement la première question de la thèse, nous nous sommes concentrés sur deux parties spécifiques du processus de conception: la résolution des problèmes sur-contraints et l'optimisation multicritère. L'idée était de concentrer l'étude de l'intelligence interactive sur une résolution de problèmes non décomposables. La résolution de problèmes sur-contraints n'est ni bien traitée par les concepteurs en raison des limitations cognitives, ni par les techniques d'intelligence artificielle. L'optimisation multicritère exige de l'intuition et de la puissance de calcul; elle est ainsi un bon exemple où la synergie entre l'homme et la machine est utile.

- Afin de rendre l'interaction la plus significative possible, nous avons du créer des visualisations adaptées à chacun des processus de conception que nous traitons. Ainsi, une des contributions scientifiques importantes de cette thèse est la création de nouveaux outils de visualisation:
  - le kaléidoscope de l'assistant Solve (voir 5.4.3),
  - la visualisation balance de l'assistant TradeOff (voir 6.3.2.6),
  - la visualisation des problèmes sur-contraints de l'assistant ElicitConflict (voir 7.3.1.2).
- Ces résultats ont abouti à des publications. Notre approche se situe quelque part entre la communauté d'intelligence artificielle et le domaine d'interaction humain-ordinateur. Nous avons publié ainsi dans les deux domaines:
  - à la conférence européenne sur l'intelligence artificielle à Budapest (ECAI, [Pu and Lalanne, 1996a]),
  - dans ≪looking to the future≫ conférence sur l'interaction humain ordinateur à Atlanta (CHI, [Lalanne, 1997]),
  - au niveau français dans les 8ème journées francophones sur l'Interaction Homme-Machine (IHM, [Lalanne, 1996]).

  Nous avons reçu dans les deux communautés des retours particulièrement encourageants.
- De nombreuses applications de notre assistant d'aide à la relaxation des problèmes sur-contraints (ElicitConflict) sont envisageables. Par exemple, un projet avec Swissair est en cours d'étude. Un utilisateur pourra, depuis le web, choisir son vol à partir d'un certain nombre de contraintes. S'il n'y a

pas de solutions, l'outil devra permettre à l'utilisateur de savoir quelles sont les contraintes en conflit ou comment relaxer son problème en étant le plus proche possible de sa définition initiale. De nombreuses autres applications pourraient être imaginables.

- Notre approche sur la relaxation des problèmes sur-contraints par les contraintes, nous semble particulièrement originale et la continuation de cette approche nous semble être la perspective la plus souhaitable.

11.3 Evaluation

Avant tout, quelques observations nous semblent nécessaires:

- Comind est plus performant qu'un humain pour résoudre des problèmes possédant des centaines de contraintes, ceci grâce à la puissance de calcul des machines et à la qualité des algorithmes de l'intelligence artificielle.
- Comind est meilleur qu'un humain pour découvrir des sous-problèmes inconsistants, pour les mêmes raisons.
- Comind permet de traiter des problèmes sur et sous-contraints. Ces deux types de problèmes mènent à des processus créatifs.
- Les humains sont plus créatifs que les machines.

Face à ces évidences, il paraît possible de dire que le système Comind augmente les facultés d'un concepteur dans une tâche créative alors que les humains enrichissent Comind de leur créativité. Cependant, la puissance de notre système serait sans valeur si le programme était incapable de communiquer avec l'utilisateur. Nous avons donc porté notre évaluation sur la facilité d'utilisation de Comind. Nous avons donné à une dizaine de concepteurs le questionnaire suivant et les avons mis face à notre système:

≪ *Comind est un système d'aide à la conception créative. Il possède différents assistants, qui sont des supports aux différentes phases de la conception. Les plus importants d'entre eux sont les suivants:*

- *L'assistant Param-Def permet de charger un problème, de le modifier ou d'en définir de nouveaux.*
- *L'assistant Eval permet d'évaluer le nombre de solutions que possède un problème.*
- *L'assistant Solve permet de trouver les solutions du problème.*
- *L'assistant TradeOff permet de visualiser les solutions et de les comparer. Il permet aussi de définir des critères afin de préciser la comparaison.*

- *L'assistant ConflictResol permet de réparer un problème qui ne possède aucune solution et de détecter les sous-problèmes inconsistants.*

Afin d'améliorer la communication entre notre système et l'utilisateur, nous demandons à des personnes comme vous de le tester. Pour que notre étude soit précise, nous vous demandons de répondre soigneusement aux questions suivantes:

1. *Pouvez-vous résoudre les problèmes suivants à l'aide du système? (de 1 à 5, 1 très mal, 5 très bien)*
   (a) *Chargez dans l'assistant Param-Def l'exemple "appartment". Utilisez ensuite l'assistant Solve pour résoudre le problème. Lorsque vous obtiendrez des solutions, utilisez l'assistant TradeOff pour les comparer et trouvez la solution optimale. Indiquez les tâches que vous avez trouvées difficiles et celles qui vous ont paru aisées.*
   (b) *Chargez dans l'assistant Param-Def l'exemple "OCS". Le problème n'a pas de solution. Utilisez l'assistant Conflict pour relaxer le problème. Cet assistant indique les contraintes les plus bloquantes. Les ensembles noirs sont des sous-problèmes inconsistants qui n'ont pas de solutions. Relaxez le problème dans l'assistant Param-Def. Puis utilisez l'assistant Solve pour trouver des solutions. Indiquez les tâches que vous avez trouvées difficiles et celles qui vous ont paru aisées.*
   (c) *Exécutez la même tâche qu'en (b) avec l'exemple "montre".*
2. *Est-ce que le système correspond à ce que vous aviez imaginé?*
3. *Combien de fonctions du système avez vous découvert? Décrivez-les brièvement.*
4. *Age: , Sexe: , Latéralité: , Formation/activité: , Langue maternelle: , Durée du test:*

Le cas (a) est un exemple de problème normalement contraint et possédant un nombre raisonnable de solutions. Le cas (b) en revanche ne possède aucune solution, c'est un cas de problème sur-contraint conflictuel. Enfin le cas (c) est un problème surdéterminé. Nous avons fait passé ce test à 6 utilisateurs, une fille et 5 garçons. Leur âge moyen était de 25 ans. L'un d'entre eux était droitier. Tous étaient des concepteurs ou des chercheurs. Les langues maternelles étaient variées (Arabe, Roumain, Français, et Allemand). Finalement, la durée moyenne du test était étonnemment courte (environ 20 minutes).

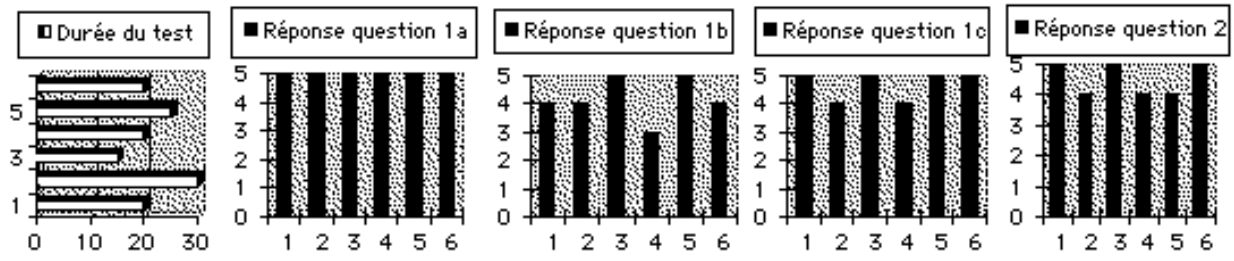

Figure 11.1: Les 6 utilisateurs étaient tous de formation scientifique.

Tous les utilisateurs ont affirmé que le système correspondait à l'idée qu'ils s'en faisaient. Il serait alors possible de dire que l'image du système correspond au modèle

du concepteur. Il nous paraît plus évident d'affirmer que les utilisateurs testés l'ont tous adopté sans problème. Aussi les utilisateurs ont tous découvert des fonctions du système qui ne leur étaient pas indiquées. L'interface les a donc invités à explorer davantage que ce qui leur était demandé. Néanmoins, en observant la manipulation des utilisateurs, nous avons trouvé des caractéristiques à améliorer. Une remarque importante est que les utilisateurs s'attendaient à ce qu'en cliquant sur l'assistant Solve, l'opération de résolution démarre. En fait, dans notre système, l'appel de cet assistant ouvre la fenêtre mais ne démarre pas l'algorithme. Il faut ensuite choisir quel algorithme démarrer. Nous pourrions commencer une recherche par défaut; si l'utilisateur n'était pas content de son choix, il pourrait le stopper et en choisir un autre. Notons que, presque chaque fois, les utilisateurs ont choisi l'algorithme random plutôt que le backtracking, pensant que le premier était plus rapide. Nous avons également constaté à quel point il est important de signaler à l'utilisateur le moment où la machine travaille, il peut ainsi se consacrer à une autre tâche de raisonnement pendant que la machine prépare ce qui lui sera nécessaire pour le faire avancer. Nous avons également été surpris de constater que les utilisateurs apprenaient rapidement une tâche que nous avions mis du temps à comprendre. Ils ont tous réussi les exercices précédents avec facilité. Ils avaient parfois des comportements inattendus. Par exemple, l'un d'eux double-cliquait sur la visualisation de l'assistant Solve afin de faire apparaître la visualisation des solutions. Naturellement, nous n'avions pas pensé à ce raccourci.

L'exercice qui a posé le plus de problèmes est le cas (c), le problème surdéterminé. Deux utilisateurs ont un moment hésité sur la contrainte à relaxer. En effet, dans cet exemple, l'utilisateur doit choisir entre 4 contraintes également responsables de l'échec de la recherche. Cependant, nous pensons que l'utilisateur, n'ayant pas défini lui-même le problème se sentait moins concerné, sa connaissance du problème étant superficielle. De plus, le temps de passation n'était en moyenne que de 5 minutes par exercice, temps trop court pour permettre à l'utilisateur d'avoir le désir profond de résoudre le problème. Nous avons constaté aussi que la visualisation des problèmes sur-contraints donnait trop de choix à l'utilisateur. Ce test nous a permis de simplifier cette visualisation dans les deux cas sur-contraints (figure 11.2 et 11.3). Nous avons donc abandonné la visualisation redondante du bas et simplifier la première. Cette nouvelle visualisation fonctionne de la même façon dans le cas surdéterminé et dans le cas sur-contraint. Seuls les plus petits conflits et leurs sous-ensembles sont affichés. Si le problème est surdéterminé, le premier lattice sera conservé intégralement puisque le plus petit conflit est le problème lui-même. Nous nous proposons donc de modifier dans le futur toutes ces caractéristiques pour que notre système soit plus facilement utilisable.

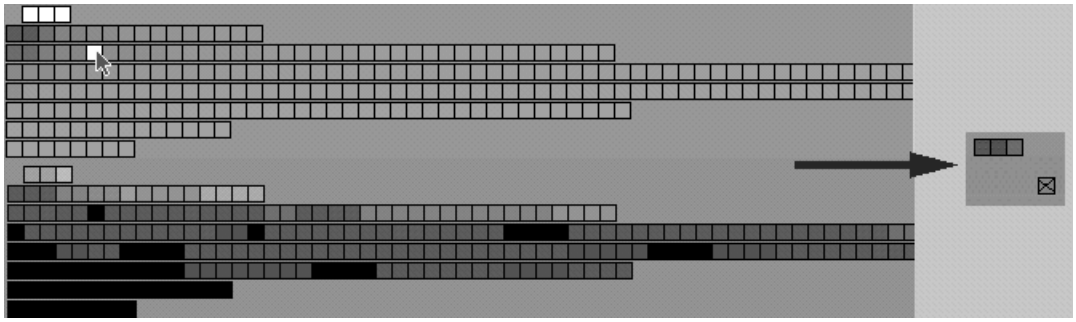

Figure 11.2: Nouvelle visualisation du conflit

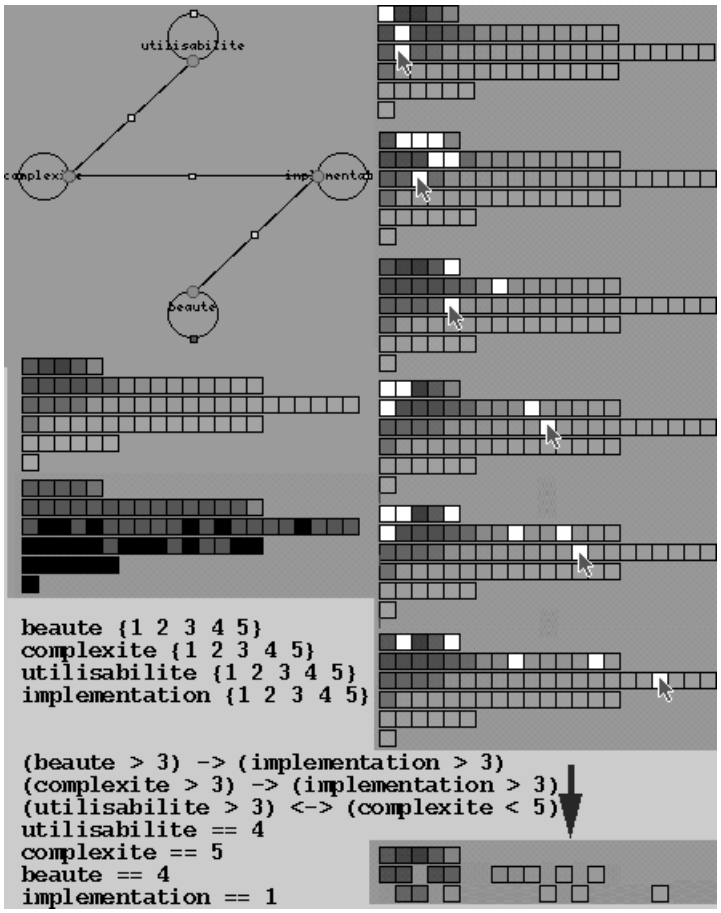

Figure 11.3: Le problème possède de multiples conflits de même taille. Certains sont disjoints et d'autres se chevauchent. Il n'y a donc pas un seul plus petit k-conflit mais plusieurs. Tous sont conservés ainsi que leurs sous-ensembles bloquants.

Petit index

Bibliographie

[Abderrahamane and al, 1994] Abderrahamane, A. and al (1994). *Eclipse 3.4, ECRC Common Logic Programming System, User Manual.*

[Baddeley, 1990] Baddeley, A. (1990). *The Human Memory, Theory and Practice.* Lawrence Erlbaum Associates.

[Bakker and al., 1993] Bakker, R. and al. (1993). Diagnosing and solving over-determined satisfaction problems. In *Proceedings the Fifth International Joint Conference on Artificial Intelligence.*

[Barbey et al., 1996] Barbey, J., Lalanne, D., and Pu, P. (1996). Evaluation de la faisabilité d'un système informatique d'aide à la recherche et à la sélection de solutions dans le cadre de la conception. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Bhatta et al., 1994] Bhatta, S., Goel, A., and Prabhakar, S. (1994). Innovation in analogical design: a model-based approach. In *Artificial Intelligence in Design*, page 57.

[Boden, 1991] Boden, M. A., editor (1991). *The Creative Mind.* Basic Books.

[Bradley et al., 1994] Bradley, S., Agogino, A., and Wood, W. (1994). Intelligent engineering component catalogs. In *Artificial Intelligence in Design*, pages 641–658. Kluwer Academic Publishers.

[Burke and Kass, 1994] Burke, R. and Kass, A. (1994). Tailoring retrieval to support case-based teaching. In *AAAI.*

[Card et al., 1983] Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates.

[Chung and Goodwin, 1994] Chung, P. and Goodwin, R. (1994). Representing design history. In *Artificial Intelligence in Design*, page 735. Kluwer Academic Publishers.

[Clavel, 1987] Clavel, R. (1987). Méthodologie de construction. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Cohen, 1981] Cohen, H. (1981). On the modelling of creative behavior. Technical report, Santa Monica, Calif.: Rand Corporation.

[Coyne et al., 1993] Coyne, R., Newton, S., and Sudweeks, F. (1993). A connectionnist view of creative design. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Cross, 1989] Cross, N. (1989). *Engineering Design Methods: Strategies for Product Design*. John Wiley and Sons Ltd.

[Dillenbourg et al., 1996] Dillenbourg, P., Baker, M., Blaye, A., and O'Malley, C. (1996). The evolution of research on collaborative learning. In *Learning in Humans and Machines*. Spada and Reimann (Eds).

[Edmonds et al., 1994] Edmonds, E., Candy, L., Jones, R., and Soufi, B. (1994). Support for collaborative design: Agents and emergence. *Communicatons of the ACM*, 37.

[Fischer, 1993] Fischer, G. (1993). Creativity enhancing design environments. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Fluckiger et al., 1996] Fluckiger, L., Piguet, L., and Baur, C. (1996). Generic robotic kinematic generator for virtual environment interfaces. *SPIE Telemanipulator and Telepresence Technologies, Boston*, 2901 Nov.

[Freuder and Wallace, 1996] Freuder, E. and Wallace, R. (1996). *Partial Constraint Satisfaction*. Lecture Notes in Computer Science. Springer-Verlag.

[Gero, 1990] Gero, J. (1990). Design prototypes: A knowledge representation schema for design. *AI Magazine*, winter 11(4).

[Gero and Maher, 1993] Gero, J. and Maher, M. (1993). Introduction of modeling creativity and knowledge-based creative design. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Giretti et al., 1994] Giretti, A., Spalazzi, L., and Lemma, M. (1994). A.s.a an interactive assistant to architectural design. In *Artificial Intelligence in Design*, page 93. Kluwer Academic Publishers.

[Gomes et al., 1996] Gomes, P., Bento, C., Gago, P., and Costa, E. (1996). Towards a case-based model for creative processes. In *European Conference on Artificial Intelligence*, page 735. Springer-Verlag.

[Gross, 1990] Gross, M. (1990). Knowledge-based subsystem layout for architectural design. In *Artificial Intelligence in Engineering - Design*. Computational Mechanics Press.

[Heath, 1993] Heath, T. (1993). Social aspects of creativity and their impact on creativity modeling. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Jampel, 1996a] Jampel, B. (1996a). *Over-Constraint Systems in CLP and CSP*. PhD thesis, City University, Belgium.

[Jampel, 1996b] Jampel, M. (1996b). *A brief overview of over-constrained systems*. Lecture Notes in Computer Science, Over-constrained systems. Springer-Verlag.

[Jussien and Boizumault, 1996] Jussien, N. and Boizumault, P. (1996). *Implementing constraint relaxation over finite domains using assumption-based truth maintenance systems*. Lecture Notes in Computer Science, Over-constrained systems. Springer-Verlag.

[Knuth, 1975] Knuth, E. (1975). Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29:121–136.

[Kumar, 1992] Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *Artificial Intelligence Magazine*.

[Lalanne, 1994] Lalanne, D. (1994). Elicitation des connaissances visuelles en medecine, un miroir de la quête mnésique de l'expert. Technical report, Diplome d'Etudes Approfondies en Sciences Cognitives, Institut National Polytechnique de Grenoble, France.

[Lalanne, 1996] Lalanne, D. (1996). Prise en compte des contraintes et des critères dans une tâche de conception. In *8ème journées sur l'Ingénierie Homme-Machine*. Cépadis ditions.

[Lalanne, 1997] Lalanne, D. (1997). Computer aided creativity and multicriteria optimization in design. In *Conference on Human Factors in Computing Systems*. Addisson Wesley.

[Lalanne, 1998] Lalanne, D. (1998). Les sytèmes d'aide à la conception créative. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Lalanne and Pu, 1996] Lalanne, D. and Pu, P. (1996). Utilisation d'icad pour la construction d'un catalogue de solutions. *Flash Informatique, Ecole Polytechnique Fédérale de Lausanne*, numéro 2.

[Laurel, 1986] Laurel, B. (1986). Interface as mimesis. In *User Centered System Design*. Donald A. Norman and Stephen W. Draper, editors, Lawrence Erlbaum Associates.

[Loga and Smithers, 1993] Loga, B. and Smithers, T. (1993). Creativity and design as exploration. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[McLaughlin, 1993] McLaughlin, S. (1993). Emergent value in creative products: some implications for creative processes. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Mitchell, 1990] Mitchell, W. (1990). *Introduction: A new agenda for computer-aided design in The Electronic Design Studio*. McCollough, Mitchell and Purcell, Eds. MIT Press, Cambridge,Mass.

[Nakakoji et al., 1994] Nakakoji, K., Summer, T., and Harstad, B. (1994). Perspective-based critiquing: helping designers cope with conflicts among intentions. In *Artificial Intelligence in Design*, page 449. Kluwer Academic Publishers.

[Nass et al., 1994] Nass, C., Steuer, J., and Tauber, E. R. (1994). Computers are social actors. In *Human Factors in Computing Systems, CHI'94 Conference Proceedings*.

[Navinchandra, 1991] Navinchandra, D. (1991). *Exploration and Innovation in Design*. Springer-Verlag, New York Inc.

[Norman, 1988] Norman, D. A. (1988). *The design of everyday things*.

[O'Sullivan, 1998a] O'Sullivan, B. (1998a). Conflict management and negotiation for concurrent engineering using pareto optimality.

[O'Sullivan, 1998b] O'Sullivan, B. (1998b). The paradox of using constraints to support creativity in conceptual design. In *Computer Aided Conceptual Design*, pages 99–121.

[O'Sullivan and Bowen, 1998] O'Sullivan, B. and Bowen, J. (1998). A constraint-based approach to supporting conceptual design. In *Artificial Intelligence in Design*, pages 291–308.

[Pareto, 1896] Pareto, V. (1896). Cours d'economie politique. Technical report, Rouge, Lausanne, Switzerland.

[Pepin, 1893] Pepin, C. (1893). Efficient design systems using case-based techniques. Technical report, University of Connecticut, Master Thesis.

[Poincaré, 1912a] Poincaré, H. (1912a). *La valeur de la Science*. Ernest Flammarion, éditeur, Paris.

[Poincaré, 1912b] Poincaré, H. (1912b). *Science et Méthode*. Ernest Flammarion, éditeur, Paris.

[Pu, 1994] Pu, P. (1994). Sigmund, would you please stop worring about the cost!: on the interaction between creative designers and intelligent machines. In *second workshop on case-based design systems, held in conjunction with the Third International Conference on Artificial Intelligence in Design*.

[Pu and Lalanne, 1996a] Pu, P. and Lalanne, D. (1996a). Human and machine collaboration in creative design. In *European Conference on Artificial Intelligence*, page 276. Springer-Verlag.

[Pu and Lalanne, 1996b] Pu, P. and Lalanne, D. (1996b). Human and machine collaboration in creative design, long version paper. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Pu and Lalanne, 1997] Pu, P. and Lalanne, D. (1997). Organization, retrieval and interaction of multimedia documents for design. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Purcell et al., 1994] Purcell, A., Gero, J., Edwards, H., and Matka, E. (1994). Design fixation and intelligent design aids. In *Artificial Intelligence in Design*, page 483. Kluwer Academic Publishers.

[Qian and Gero, 1992] Qian, L. and Gero, J. (1992). A design support system using analogy. In *Artificial Intelligence in Design*, page 795. Kluwer Academic Publishers.

[Radford and Gero, 1988] Radford, A. and Gero, J. (1988). *Design by optimization in architecture, building, and construction.* Van Nostrand Reinhold Company, New York.

[Riesbeck and Schank, 1989] Riesbeck, C. and Schank, R. (1989). *Inside case-based reasoning.* Lawrence Erlbaum Associates.

[Rolland, 1998] Rolland, L. (1998). Conception de mécanismes, élaboration des principes de mobilité. Technical report, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[Rosenman and Gero, 1993] Rosenman, M. and Gero, J. (1993). Creativity in design using a design prototype approach. In *Modeling Creativity and Knowledge-Based Creative Design.* J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Sharp, 1995] Sharp, J. (1995). Ai system support for conceptual design. In *Proceedings of the 1995 Lancaster International Workshop on Engineering Design.* Springer.

[Shneiderman, 1990] Shneiderman, B. (1990). *Designing the User Interface, Strategies for effective Human-Computer Interaction.* Addison Wesley.

[Spence et al., 1995] Spence, R., Colgan, L., and Rankin, P. (1995). The cockpit metaphor. *Behaviour and Information Technology*, 14, 4:251–263.

[Sun and Faltings, 1994] Sun, K. and Faltings, B. (1994). Supporting creative mechanical design. In *Artificial Intelligence in Design*, page 39. Kluwer Academic Publishers.

[Takala, 1993] Takala, T. (1993). A neuropsychologically-based approach to creativity. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Tomlin, 1986] Tomlin, D. (1986). Personnal communication. Technical report, Harvard University.

[Tweedie et al., 1996] Tweedie, L., Spence, R., Dawkes, H., and Su, H. (1996). Externalising abstract mathematical models. In *Conference on Human Factors in Computing Systems*. Canada, ACM Press.

[Tweedie et al., 1994] Tweedie, L., Spence, R., Williams, D., and Bhoghal, R. (1994). The attribute explorer. In *Conference on Human Factors in Computing Systems, Video Proceedings*. Mass. ACM Press.

[Weisberg, 1993] Weisberg, W. (1993). *Creativity, beyond the myth of genius*. W.H. Freeman and Company, New York.

[Wills and Kolodner, 1994] Wills, L. M. and Kolodner, J. L. (1994). Towards more creative case-based design systems. In *Proceedings of National Conference on Artificial Intelligence*.

[Woodbury, 1993] Woodbury, R. (1993). A genetic approach to creative design. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

[Woolf and Hall, 1995] Woolf, B. and Hall, W. (1995). Multimedia pedagogues, interactive systems for teaching and learning. *IEEE Computer*, May:74–80.

[Zhao and Maher, 1992] Zhao, F. and Maher, M. (1992). Using network-based prototypes to support creative design by analogy and mutation. In *Modeling Creativity and Knowledge-Based Creative Design*. J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers.

Curriculum Vitae

Lalanne Denis
1st of March 1971
French
Single

1991 Degree in Science (Math, Physics, Chemistry, Computers)
DEUG Sciences et Structures de la Matiere, Universite Joseph Fourier (UJF), Grenoble

1993 Bachelor Degree in Computer Science
Licence + Maitrise en Informatique, Institut des Mathematiques Appliquees de Grenoble (IMAG)

1994 Degree in Social Sciences (Psychology, Sociology, Biology)
DEUG Mathematiques Appliquees aux Sciences Sociales, Universite Pierre-Mendes-France, Grenoble (UPMF)

1994 Master Degree in Cognitive Science
Diplome d'Etudes Approfondies en Sciences Cognitives, Institut National Polytechnique de Grenoble (INPG)

1995-1998 Research assistant and Ph.D. student in the Ergonomics of Intelligent Systems and Design Laboratory, ISR/DMT, Ecole Polytechnique Fédérale de Lausanne.