Evolution + Adaptation = Résolution

Vincent Barichard, Hervé Deleau, Jin-Kao Hao, Frédéric Saubion

Evolution + Adaptation = Résolution

Vincent Barichard, Hervé Deleau, Jin-Kao Hao et Frédéric Saubion
LERIA, Université d'Angers
2, Bd Lavoisier, 49045 Angers Cedex 01
{barichar,deleau,hao,saubion}@info.univ-angers.fr

Résumé

Nous présentons dans cet article les bases d'un nouveau modèle de calcul permettant de combiner des méthodes complètes et incomplètes pour la résolution de problèmes de satisfaction de contraintes. Ce schéma algorithmique utilise des techniques de propagation de contraintes dans un contexte évolutionniste intégrant également des heuristiques de recherche locale. L'uniformité des structures utilisées autorise une interaction plus homogène entre les différentes méthodes mises en oeuvre et permet également de bénéficier au mieux de leurs atouts respectifs. La grande flexibilité de ce modèle offre également la possibilité d'en envisager diverses extensions. Nous mettons en avant l'intérêt de notre approche sur quelques exemples par le biais d'une implémentation.

Mots-clés : Problèmes de Satisfaction de Contraintes - Algorithmes évolutionnistes - Propagation de contraintes - Recherche locale

1 Introduction

Les problèmes de satisfaction de contraintes [15] (Constraint Satisfaction Problems CSP) permettent de modéliser un grand nombre de problèmes pratiques (plannification, ordonnancement, emploi du temps ...). Ils sont caractérisés par un ensemble de variables pouvant prendre leurs valeurs dans des domaines particuliers et un ensemble de contraintes. Les contraintes sont des relations entre les variables et la résolution du problème consiste à trouver une combinaison de valeurs satisfaisant ces contraintes. De nombreux algorithmes et systèmes ont été développés pour résoudre les CSP. On distinguera au moins deux grands types d'approches qui diffèrent à la fois dans leur finalité et dans les méthodes qu'elles mettent en oeuvre.

— Les *approches complètes* sont capables d'explorer souvent de manière implicite, l'intégralité de l'espace de recherche pour en extraire toutes les solutions ou bien, le cas échéant, pour détecter l'absence de solution. Cette complétude est toutefois obtenue au détriment d'un coût calculatoire important, ce qui limite bien souvent leur utilisation dans le cadre des problèmes de grande taille. Ces méthodes utilisent principalement la

notion de consistance locale [11, 13] qui permet de réduire l'arbre d'exploration en élagant des branches ne contenant pas de solution.

— Les *approches incomplètes* quant à elles, reposent essentiellement sur l'utilisation d'heuristiques permettant de parcourir plus efficacement certaines zones de l'espace de recherche afin d'y trouver éventuellement des solutions. Alors que cette démarche privilégie une efficacité calculatoire, elle ne permet en aucun cas de garantir l'obtention de telles solutions et encore moins de constater leur absence.

Un principe largement partagé pour la conception d'algorithmes toujours plus performants et robustes, consiste à tirer parti des avantages respectifs des différentes approches de résolution d'un problème et conduit donc souvent à les combiner.

L'hybridation des deux approches décrites plus haut a déjà été étudiée dans le cadre de la résolution de CSP [8, 9, 16]. Mais, souvent, cette hybridation se révèle relativement hétérogène dans sa formulation, méthodes complètes et incomplètes utilisant des structurations différentes de l'espace de recherche.

Du côté des méthodes incomplètes, divers algorithmes hybrides combinant approches évolutionnistes et recherche locale ont été utilisées pour la résolution de CSP particuliers (par exemple [6]).

Notre objectif est de proposer un modèle plus uniforme permettant d'intégrer ces divers éléments. Nous développons pour celà un cadre évolutionniste basé sur le concept d'adaptation d'un ensemble d'individus à un environnement.

Résoudre un CSP consiste à parcourir un ensemble d'affectations possibles de valeurs aux variables afin d'en extraire des solutions. Nous pouvons considérer cet ensemble comme une population d'individus qui s'adapte à deux critères principaux : la consistance locale qui permet de reduire la taille des zones à explorer et la consistance globale qui consiste à se rapprocher d'une solution dans une zone donnée. Nous disposerons d'une population permettant de couvrir en permanence les zones contenant des solutions et qui évoluera de manière à permettre l'exploration de zones de plus en plus précises. Cette population sera utilisée pour générer un sous ensemble d'individus sur lesquels nous intensifiront la recherche et qui évoluera donc vers de possibles solutions. Ces différentes adapations et évolutions se font par l'application d'opérateurs agissant sur les populations et leurs individus.

Dans la section suivante, nous rappellerons les concepts de bases relatifs aux CSP. La section 3 formalise l'ensemble des éléments utilisés dans notre approche. Nous en donnerons une description générale mettant en avant les interactions essentielles entre ces composants. La section 4 présentera une instanciation du schéma précédent aux travers d'une implémentation permettant de valider notre approche sur quelques exemples.

2 Problèmes de Satisfaction de Contraintes

Nous proposons ici une formulation générale des problèmes de satisfaction de contraintes (CSP) [15]. On se donne un ensemble de variables $X = \{x_1, \cdots, x_n\}$

dont les domaines respectifs sont dans l'ensemble $D = \{D_1, \cdots, D_n\}$. Une contrainte quelconque est une relation $c \subseteq D_1 \times \cdots \times D_n$. Un CSP est donc classiquement défini par un triplet (X, D, C) où C est un ensemble de contraintes. Une valuation est une fonction $v \colon X \to \bigcup_{1 \leq i \leq n} D_i$ telle que $\forall 1 \leq i \leq n, v(x_i) \in D_i$. Une valuation v est une solution du CSP (X, D, C) si et seulement si $\forall c \in C, (v(x_1), \cdots, v(x_n)) \subseteq c$. Nous noterons Sol l'ensemble des solutions du CSP.

3 Un algorithme évolutionniste pour les CSP

Nous définissons un cadre évolutionniste uniforme permettant d'intégrer propagation de contraintes et méthodes heuristiques dans un schéma algorithmique commun. Les algorithmes évolutionnistes [7] sont principalement basés sur la notion d'adaptation d'une population d'individus à un critère par le biais d'opérateurs tels que le croisement [5].
Notre méthode opère sur des populations d'individus représentant des portions de l'espace de recherche du CSP initial, ces portions pouvant être réduites à des points lorsque l'on cherchera des solutions.
Elle repose ensuite sur deux phases d'adaptation successives. Une première phase d'adaptation locale a pour but de faire évoluer une population en fonction d'un critère de consistance locale par rapport aux contraintes du CSP. La deuxième phase d'adaptation globale travaille sur une sous population issue de la population précédente et dont le but est de s'adapter à un critère de satisfaction global des contraintes. Nous utiliserons donc deux populations d'individus, l'une permettant de couvrir l'espace de recherche (que nous appellerons population de couverture), l'autre nous permettant d'intensifier la recherche dans certaines zones afin de trouver des solutions (appellée population d'intensification).
Des opérateurs de sélection et d'évolution permettent de modéliser les diverses stratégies de recherche favorisant l'une ou l'autre des phases. Le schéma général de notre méthode est présenté figure 1.

3.1 Représentation et Espace de Recherche

Pour un CSP (X, D, C), une approche classique pour les méthodes évolutionnistes consiste à considérer comme espace de recherche l'ensemble des valuations possibles qui peut donc être assimilé à l'ensemble des n-uplets $S = D_1 \times \cdots \times D_n$ de valeurs possibles pour les variables.
Ici nous allons utiliser des individus qui représentent en fait une partie de l'espace de recherche précédent en associant à chaque variable non plus une valeur mais un sous ensemble de son domaine.

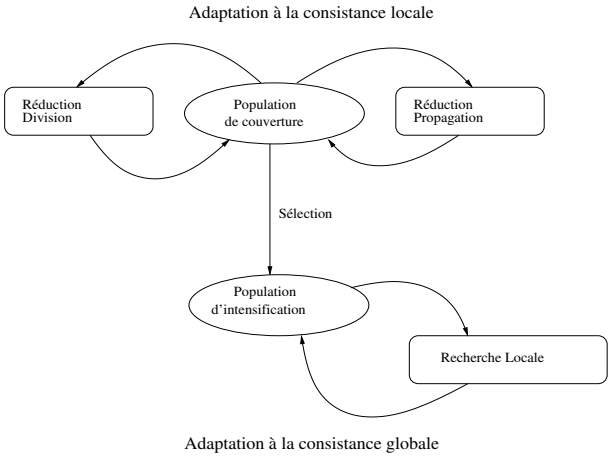

FIGURE 1 – Principe général

Définition 1 Individu
Etant donné un CSP (X, D, C)*, l'ensemble des individus est défini par :*

$$I = \{(d_1, \cdots, d_n) \mid \forall\ 1 \leq i \leq n, d_i \subseteq D_i\}$$

Pour tout $\xi \in I$, ξ_i désigne la $i^{ème}$ composante de ξ. Nous en déduisons immédiatement que $\xi \in I$ est une solution ssi $\forall\ 1 \leq i \leq n, |\xi_i| = 1$ et $\forall\ c \in C, \xi \subseteq c$. ξ est inconsistant ssi $\exists 1 \leq i \leq n, \xi_i = \emptyset$.
Nous définissons un ordre $\sqsubseteq$ sur les individus comme suit.

Définition 2 Ordre sur les individus

$$\forall\ \xi, \xi' \in I, \xi \sqsubseteq \xi'\ \ ssi\ \forall\ i, \xi_i \subseteq \xi'_i$$

Remarquons que, selon cet ordre, plus un individu est petit, plus la zone de l'espace de recherche recouverte est réduite.
Nous obtenons un treillis $(I, \sqsubseteq)$ qui possède un plus petit élément $(\emptyset, \cdots, \emptyset)$ et un plus grand élément $(D_1, \cdots, D_n)$.
Un individu qui vérifie $\forall\ 1 \leq i \leq n, |\xi_i| = 1$ sera appelé un point. La notion de solution d'un CSP définie section 2 concerne donc les points.
Pour un individu $\xi = (d_1, \cdots, d_n)$ nous noterons $point(\xi) = \{\xi' \in I \mid \forall\ 1 \leq i \leq n, |\xi'_i| = 1 \wedge \xi' \subseteq \xi\}$ l'ensemble des points contenus dans ξ. Nous noterons donc $Sol(\xi) = point(\xi) \cap Sol$ l'ensemble des solutions contenues dans ξ.
Une population est un ensemble d'individus et donc l'ensemble $P = 2^I$ des populations est l'ensemble des parties de I. L'ensemble des solutions contenues dans une population $p \in P$ sera $Sol(p) = \bigcup_{\xi \in p} Sol(\xi)$. Nous étendons l'ordre précédent à P.

Définition 3 Ordre sur les populations

$$\forall\ p, p' \in P, p \sqsubseteq p'\ \ ssi\ \forall\ \xi \in p, \exists \xi' \in p' \xi \sqsubseteq \xi'$$

Nous avons donc un semi-treillis $(P, \sqsubseteq)$ avec un plus petit élément $\emptyset$.
Notre espace de recherche est donc à présent ce semi-treillis des populations dans lequel nous allons nous déplacer selon deux grands principes. Le premier de ces principes consiste à adapter une population au critère de consistance locale.

3.2 Adaptation d'une population à la notion de consistance locale

La notion de consistance locale [17] permet, par un examen individuel des contraintes, d'éliminer des points qui ne sont pas solutions. Cette notion n'implique pas la consistance globale mais la réciproque est vraie. L'adaptation de la population de couverture à la consistance locale a pour but d'obtenir une population couvrant une zone de plus en plus réduite de l'espace de recherche mais conservant les mêmes solutions. Pour réduire progressivement cette population nous allons utiliser deux opérations de base : la réduction des domaines et la division des domaines.

3.2.1 Réduction par propagation

Inspirés par les travaux de [2, 1], nous présentons ici la propagation de contraintes comme un ensemble d'opérateurs de réduction de domaines permettant de calculer un point fixe. Nous définissons dans un premier temps cette notion d'opérateur sur les individus.

Définition 4 *Etant donné un CSP (X, D, C), un opérateur de propagation pour une contrainte $c \in C$ est une fonction $\pi : I \to I$ qui vérifie les propriétés suivantes :*
- $\pi(\xi) \sqsubseteq \xi$ *(contraction)*
- $\xi \cap c \sqsubseteq \pi(\xi)$ *(correction)*
- $\xi \sqsubseteq \xi' \Rightarrow \pi(\xi) \sqsubseteq \pi(\xi')$ *(monotonie)*

Notons *red* l'ensemble des opérateurs de réductions associés au CSP et vérifiant la définition précédente. Nous savons d'après [2] que l'application successive d'un ensemble *red* de tels opérateurs nous permet à partir d'un individu d'obtenir un point fixe qui correspond à un individu plus petit (au sens de $\sqsubseteq$) qui contient les mêmes solutions.
Dans notre contexte, la propagation de contraintes s'opère sur une population d'individus et non plus simplement sur un n-uplet de domaine.
Afin de rester le plus général possible et d'abstraire toute stratégie d'application vis à vis des différents individus d'une population, nous étendons de manière naturelle cette notion d'opérateurs à une population. Nous considérons donc un ensemble d'opérateurs :

$$\Pi : P \times red \to P$$
$$(p, \pi) \mapsto p_1 \cup \{\pi(\xi) \mid \xi \in p_2\} \text{ avec } p = p_1 \cup p_2 \wedge p_1 \cap p_2 = \emptyset$$

Cette généralisation immédiate indique simplement que l'on peut choisir d'appliquer simultanément des réductions sur plusieurs individus en choisissant une partition p_1, p_2 de p. Au niveau des stratégies possibles, nous pouvons par exemple distinguer l'application d'un opérateur de propagation à tous les individus d'une même population ou bien à un individu particulier. Nous appelons RED l'ensemble de ces opérateurs.
Cette définition permet donc de réduire par propagation tous les individus d'une population p jusqu'à obtenir un point fixe en itérant un ensemble d'opérateurs RED. Nous notons p_{RED} ce point fixe. Nous devons bien sûr éliminer de p_{RED} tous les individus qui ne sont pas consistants localement et donc globalement. Ces individus $\xi \in p_{RED}$ sont tels que : $\exists 1 \leq i \leq n, \xi_i = \emptyset$.
Nous définissons à partir de la réduction précédente une fonction sur P.

Définition 5 Consistance locale

$$\begin{array}{l} CL\colon P \to P \\ p \mapsto \{\xi \in p_{RED} \mid \forall\ 1 \leq i \leq n, \xi_i \neq \emptyset\} \end{array}$$

Nous remarquons bien évidemment que $CL(p) \sqsubseteq p$ et que les solutions sont conservées par cette réduction ($Sol(p) \subseteq Sol(CL(p))$) (puisque selon la définition 4, les opérateurs basiques de réduction conservent les solutions). Cette fonction va nous permettre de nous déplacer sur le semi-treillis des populations en passant d'une population à une population plus petite contenant les mêmes solutions. Nous devons maintenant définir un opérateur permettant de réduire encore les sous espaces de recherches couverts par la population.

3.2.2 Réduction par division

Nous utilisons ici un opérateur d'évolution particulier qui diffère des opérateurs de croisements [5] usuels. Nous définissons cet opérateur dans un premier temps sur les individus. Il s'agit d'une division cellulaire dans laquelle un individu seul va générer deux nouveaux individus. Cette division est contrôllée par la donnée d'une variable dont le domaine sera découpé.
Nous définissons donc un ensemble d'opérateurs pour les variables $x \in X$. Pour un individu $\xi \in I$ et une variable $x \in X$, nous notons ξ_x le domaine de la variable x dans ξ. Notre opérateur sera donc indexé par une variable.

$$\begin{array}{l} \chi_x\colon I \to I \times I \\ \xi \mapsto (\xi^1, \xi^2) \\ \quad \text{avec } \xi^1_x \cap \xi^2_x = \emptyset \wedge \xi^1_x \cup \xi^2_x = \xi \\ \forall\ y \neq x, \xi^1_y = \xi^2_y = \xi_y \end{array}$$

Nous notons χ_X l'ensemble de ces opérateurs. Nous remarquons que $\xi^1 \sqsubseteq \xi$ et $\xi^2 \sqsubseteq \xi$. Nous avons la propriété immédiate :

Propriété 1 *χ_x conserve les solutions : $Sol(\xi) = Sol(\xi^1) \cup Sol(\xi^2)$.*

Nous généralisons cet opérateur pour une population afin d'autoriser plusieurs divisions simultanées.

$$\chi_x : P \to P$$
$$p \mapsto p_1 \cup \{\chi_x(\xi) \mid \xi \in p_2\} \text{ avec } p = p_1 \cup p_2 \wedge p_1 \cap p_2 = \emptyset$$

Une stratégie de division consiste juste à définir p_1 et p_2 ainsi qu'à choisir la variable x fournissant l'opérateur χ_x. Pour cela nous introduisons deux exemples d'opérateurs de sélection des individus qui vont nous permettre de mettre en oeuvre deux stratégies de parcours.

$$\sigma_1 : P \to P$$
$$p \mapsto \{min(p)\}$$

où $min(p)$ renvoit un individu ξ dont les tailles des domaines sont minimaux c'est à dire vérifiant $\forall\ \xi' \in p, \xi' \neq \xi \forall\ 1 \leq i \leq n, |\xi_i| \leq |\xi'_i|$. Cette fonction σ_1 nous permet de concentrer la recherche sur des individus dont les domaines auront des tailles de plus en plus petites. Nous pouvons également simuler un parcours en largeur.

$$\sigma_2 : P \to P$$
$$p \mapsto \{\xi \in p \mid \nexists \xi' \in p, \xi' \sqsubset \xi\}$$

Nous notons Σ l'ensemble des opérateurs de sélection. Suivant notre principe général, cette selection pourra être utilisée pour sélectionner les candidats à une division cellulaire ainsi que les individus à explorer dans une deuxième phase.
Nous pouvons à présent définir notre succession d'adaptation-évolution pour la consistance locale comme la donnée d'une suite de couples $(\chi^i, \sigma^i)_{1 \leq i \leq k}$ avec $\chi^i \in \chi_X$ et $\sigma^i \in \Sigma$. Etant donnée une population $p \in P$ nous obtenons son adaptation comme la suite :

$$p_0 = CL(p)$$
$$p_{i+1} = CL((p_i \setminus \sigma^i(p_i)) \cup \chi^i(\sigma^i(p_i)))$$

Nous noterons $ACL(p, (\chi_i, \sigma_i)_{1 \leq i \leq k}) = p_k$ la population obtenue après l'itération de k processus *consistance-sélection-division*.
Il nous faut à présent préciser la manière dont nous utilisons les individus de la population de couverture pour générer la population d'intensification.

3.3 Sélection de la sous population pour la recherche intensive de solutions

Nous nous intéressons ici à la préparation d'une population dont le but sera de s'adapter à un critère de consistance globale c'est à dire de générer des individus solutions ou, en tout cas, les plus proches possible d'une solution. Notre cadre est suffisament général pour appliquer des méthodes de recherche locale qui travailleraient sur des individus quelconques, toutefois nous allons définir la suite de notre modèle en ne considérant à ce niveau que des points. Nous définissons

un opérateur de génération de points à partir d'un individu de la population de couverture. Afin par la suite de pouvoir guider la recherche locale en fonction de la zone à explorer et donc de l'individu à l'origine des points, nous mettons en indice sur chaque individu de la population d'intensification, l'individu de la population de couverture dont il est issu.

$$\begin{array}{l}\gamma\colon P \times \Sigma \to P \\ (p,\sigma) \mapsto \gamma(p,\sigma) \text{ avec } \gamma(p,\sigma) \subseteq \{\xi'_\xi \in I \mid \xi \in \sigma(p), \xi'_\xi \in point(\xi)\}\end{array}$$

Ce sont donc les individus de la population de couverture qui vont canaliser l'intensification sur les points qu'ils permettent de générer et l'indexation qui est pratiquée permet d'attacher son "père" à chaque point de la population d'intensification $\gamma(p,\sigma)$.

3.4 Adaptation à la consistance globale

Le but de cette deuxième phase est d'adapter la population d'intensification à un critère de consistance globale en rapport direct avec la notion de solution. De manière habituelle, ce critère peut être formalisé par l'ensemble des contraintes non satisfaites par un individu. Un individu sera donc d'autant meilleur qu'il minimise ce critère. Nous utiliserons la recherche locale pour réaliser cette adaptation. Notre objectif est de présenter les principaux composants d'une recherche locale sans entrer dans ses détails algorithmiques.
Nous définissons donc la fonction d'évaluation suivante :

Définition 6 Evaluation

$$\begin{array}{l}\psi\colon I \to C \\ \xi \mapsto \{c \in C \mid \xi \notin c\}\end{array}$$

Nous avons bien évidemment la propriété :

Propriété 2

$$\forall\, \xi \in I, \psi(\xi) = \emptyset \Rightarrow \xi \in Sol$$

Nous devons à présent, selon le principe fondateur des méthodes dites de recherche locale, pouvoir nous déplacer de proche en proche parmi les points afin de rechercher des éléments toujours plus adaptés au critère précédent. Nous avons vu que nous associons à chaque point de la population un indice correspondant à l'individu de la population de couverture dont il est issu. De fait, nous allons définir pour des points ξ'_ξ des opérateurs ω_ξ qui seront restreints par ξ. Un tel opérateur aura le profil suivant :

$$\begin{array}{l}\omega_\xi\colon I \to I \\ \xi'_\xi \mapsto \omega_\xi(\xi'_\xi)\end{array}$$

avec $\omega_\xi(\xi'_\xi) \subseteq point(\xi)$. Nous voyons donc clairement le rôle de ξ qui confine l'opérateur dans la zone de l'espace de recherche qu'il représente.
Nous nous donnons d'abord une fonction de voisinage.

$$\mu_\xi : I \to P$$
$$\xi'_\xi \mapsto \mu(\xi'_\xi) \text{ avec } \forall \xi''_\xi \in \mu(\xi'_\xi), \xi''_\xi \in point(\xi)$$

Il nous reste maintenant à déterminer des stratégies d'exploration de ces voisinages. Pour cela, nous allons définir deux exemples d'opérateurs de base sur les individus permettant de modéliser diverses stratégies de recherche locale. La première stratégie basique (appellée couramment descente) consiste à rechercher des voisins toujours meilleurs. Nous souhaitons toujours nous déplacer dans un espace de recherche limité et nous choisissons que cet opérateur s'arrête lorsqu'une solution est atteinte.

$$D_\xi : I \to P$$
$$\xi'_\xi \mapsto \begin{cases} \xi'_\xi \text{ si } \xi'_\xi \in Sol \\ D_\xi(\xi'_\xi) \subseteq \{\xi''_\xi \in I \mid \xi''_\xi \in \mu_\xi(\xi'_\xi), \psi(\xi''_\xi) < \psi(\xi'_\xi)\} \text{ sinon} \end{cases}$$

Cet opérateur s'étend naturellement à une population : $D_\xi(p) = \bigcup_{\xi'_\xi \in p} D_\xi(\xi'_\xi)$.
A partir d'un individu ξ'_ξ quelconque, un tel opérateur atteint toujours un point fixe qui sera un ensemble d'optima locaux. Nous aurons donc :

Propriété 3

$$\exists k \in I\!N, D_\xi(D^k_\xi(\xi'_\xi)) = D^k_\xi(\xi'_\xi)$$

Cette propriété découle naturellement de l'ordre sur I induit par la fonction ψ. Afin de pouvoir diversifier notre recherche et sortir de cet ensemble s'il ne contient pas de point solution, nous introduisons un autre opérateur autorisant des chemins alternatifs dans le voisinage.

$$CA_\xi : I \to P \times I$$
$$\xi'_\xi \mapsto \begin{cases} \xi'_\xi \text{ si } \xi'_\xi \in Sol \\ CA_\xi(\xi'_\xi) \subseteq \{\xi''_\xi \in I \mid \xi''_\xi \in \mu_\xi(\xi'_\xi)\} \text{ sinon} \end{cases}$$

Une recherche locale rl_ξ est une séquence finie de symboles de $\{D_\xi, CA_\xi\}$.
Etant donné une recherche locale rl_ξ et un individu ξ'_ξ, nous noterons $rl_\xi(\xi'_\xi)$, la population obtenue par applications successives des opérateurs de rl_ξ. Nous noterons RL l'ensemble des recherches locales.
D'un point de vue pratique, l'implémentation de tels opérateurs consiste à choisir comme sous ensemble généré à chaque itération un seul point selon une stratégie particulière (le premier point de l'ensemble, un point choisi aléatoirement dans l'ensemble ...). De plus, nous remarquons que cette définition générique permet de considérer différents algorithmes classiques pour la recherche locale, une stratégie étant essentiellement caratérisée par la manière d'alterner les D_ξ et les CA_ξ. Cette alternance peut être controllée par un paramètre aléatoire comme dans le recuit simulé [10], par une mémorisation des états précédents comme dans la méthode tabou [4] ou par toute autre heuristique.
Nous pouvons à présent définir notre fonction d'adaptation à la consistance globale qui applique donc un ensemble de recherches locales adaptées au individus de la population d'intensification.

$S = \emptyset$
$p \leftarrow \{(D_1, \cdots, D_n)\}$
Repeter
    Choisir $\sigma \in \Sigma, x \in X$
    $p = ACL(p, (\chi_x, \sigma))$
    $S = S \cup (p \cap Sol)$
    $p = p \setminus (p \cap Sol)$
    Choisir $\sigma \in \Sigma, \rho \in 2^{RL}$
    $p' = ACG(\gamma(p, \sigma), \rho)$
    $S = S \cup (p' \cap Sol)$
Jusqu'a *stop*

FIGURE 2 – Algorithme général

Définition 7

$$ACG\colon P \times 2^{RL} \to P$$
$$(p, \rho) \mapsto \bigcup_{\xi'_\xi \in p} rl_\xi(\xi') \ avec \ rl_\xi \in \rho$$

3.5 Algorithme Général

Nous présentons ici l'algorithme général dans une version simplifiée où nous opérons successivement une adaptation de chaque type. Nous introduisons un ensemble permettant de collecter les solutions trouvées au cours du processus global. Nous devons remarquer qu'une solution peut apparaitre soit lors de la phase d'adaptation à la consistance locale, après une division qui aura produit un point solution, soit lors de la phase d'adaptation à la consistance globale, au cours de la recherche locale. Cet algorithme est présenté figure 2.
Le critère d'arrêt *stop* mentionné ici peut être vu de différentes manières. On peut choisir de s'arrêter lorsque la population de couverture p est vide, auquel cas on aura couvert tout l'espace de recherche et donc trouvé toutes les solutions. On peut choisir de stopper lorsque la première solution est trouvée ou bien lorsqu'un certain nombre de solutions ont été construites. Enfin, on peut également limiter le processus à un certain nombre d'itérations. Ces divers choix nous permettent d'occiler entre algorithme complet et incomplet.
On voit que la réponse fournie par l'algorithme est double. D'une part, on dispose de l'ensemble S des solutions trouvées et, d'autre part, on dispose également de la population p qui nous donne la couverture courante de l'ensemble des solutions.

3.6 Extensions et particularités

Nous discutons ici des diverses extensions et spécificités de notre modèle de calcul.
Taille de la population de couverture

La taille maximale de la population de couverture permet de contrôler et de limiter la progression dans le développement des zones de recherche. Les opérateurs de sélection permettent ensuite de gérer les réductions par division. Si une taille maximale est fixée, les découpes s'arrêteront, laissant place à la recherche locale. On supprime des individus de cette population de couverture dans deux cas : lorsqu'ils correspondent à des points solutions ou lorsqu'ils ne sont pas consistants localement. Nous pouvons envisager également, par exemple, de perdre la complétude de la couverture en supprimant des individus lorsqu'une solution a été trouvée par l'intensification dans la zone de recherche qu'ils représentent (afin d'extraire des solutions dispersées). Ceci met en avant clairement que la population de couverture permet par son évolution de guider la recherche locale dans la phase d'intensification.

Problème Max-CSP

D'une manière naturelle notre cadre traite le problème Max-CSP qui consiste à satisfaire le plus de contraintes possible d'un CSP donné. Il nous suffit pour cela de mémoriser lors de la phase d'adaptation à la consitance globale, la meilleure solution trouvée du point de vue de la fonction d'évaluation ψ.

Problèmes d'optimisation sous contraintes

Ce cadre général peut s'étendre aux problèmes d'optimisation sous contraintes dans lesquels on doit optimiser une fonction objective tout en satisfaisant l'ensemble des contraintes. On peut dans ce cas, utiliser la fonction objective afin de calculer une estimation de sa valeur pour les individus de la population de couverture. On peut alors en éliminer les individus dont la valeur objective estimée n'est pas intéressante. Ceci montre que les méthodes d'élagage de type *branch and bound* s'intègrent naturellement à notre schéma général.

Introduction d'opérateurs de recombinaison et extension de la recherche locale

Tirant partie de la structure de notre population d'intensification nous pourrions étendre les mécanismes de recherche utilisés en introduisant des processus de recombinaison d'individus à ce niveau. Il faut contrôler que ces processus ne nous font pas sortir de l'espace délimité par l'individu pour l'intensification et qui restreint le voisinage ou bien gérer cette sortie. Il est également possible d'envisager une recherche locale sortant de la zone où elle est normalement confinée. Dans ce cas, on peut, par exemple, faire intervenir l'éloignement par rapport à cette zone dans la fonction d'évaluation pour contrôler cette évolution.

Algorithmique distribuée

Le modèle que nous proposons ici est parfaitement adapté à une algorithmique distribuée. En effet, les populations peuvent être réparties sur plusieurs processeurs et traitées séparément, que ce soit au niveau de la population de couverture ou au niveau des populations d'intensification.

3.7 Travaux connexes

De nombreux travaux ont été menés sur l'hybridation possibles des méthodes à base de recherche locale et des techniques de propagation de contraintes. Il ne serait pas possible ici de faire un tour complet d'horizon de ces propositions

dans la mesure où bien souvent elles correspondent à un algorithme spécifique dédié à un problème particulier.
Nous pouvons mentionner [3] qui présente un panorama général de l'utilisation possible de la recherche locale dans le cadre de la programmation par contraintes. D'autres travaux [9, 8] visent à uniformiser la combinaison de ces méthodes et leurs principaux aspects.
Nous pouvons également mentionner [16] comme application d'hybridation de méthode d'arc consistance pour guider une recherche tabou appliquée à un problème de type Max-csp. [14] propose également une combinaison de techniques pour améliorer l'efficacité des méthodes évolutionnistes dans le cadre de la résolution de CSP.

4 Application

Nous présentons dans cette section une implémentation des concepts décrits plus haut. Ce prototype nous permet de valider notre approche sur quelques exemples de CSP.

4.1 Choix d'implémentation

Nous avons implémenté des opérateurs de propagation de contraintes basés sur la consistance de bornes (voir [12]) ainsi qu'un traitement spécifique des diséquations. La division des domaines est effectuée en coupant au milieu le plus grand domaine de l'individu minimal. Concernant la phase d'intensification, nous avons implémenté une simple descente avec un chemin aléatoire. Nous introduisons un point dans la population d'intensification pour chaque individu de la population de couverture.
Nous comparons les performances obtenues par chacune des deux phases utilisées de manière indépendante avec celles obtenues par la combinaison de ces deux phases. Les deux phases de l'algorithme utilisées séparément simulent respectivement une méthode de résolution complète avec backtracking et consistance locale à chaque noeud de l'arbre et une recherche locale.

4.2 Présentation des problèmes

Notre but n'est pas ici d'adapter notre modèle pour une résolution optimisée d'un problème spécifique de grande taille mais plutôt de prouver la validité de notre approche sur un échantillon de problèmes simples et variés (au niveau du nombre de variables, des contraintes et des domaines).
S+M=M
Il s'agit d'un problème de crypto-arithmétique consitant à résoudre l'équation $SEND + MORE = MONEY$. Il comporte 8 variables, une contrainte d'égalité et de nombreuses diséquations indiquant que les variables sont distinctes deux à deux. Les domaines possèdent 10 valeurs.

Problème	ACL [1] nb ind. gén.	ACL + ACG (nb ind. gén,nb mvts)	ACG [2] (tx succès,nb mvts)
S+M=M	3111	(3071,15358)	(0%,-)
Sac	-	(8,202)	(100%,36)
Affectation-1	14	(4,11)	(100%,214)
Affectation-2	34	(8,44)	(100%,1480)
Ordo	218	(2,9)	(100%,14)

TABLE 1 – Obtention d'une solution (ACL vs. ACL+ACG vs. ACG)

Sac
Il s'agit d'un problème de sac à dos avec 30 variables ayant des domaines de taille 20 et 3 contraintes. Il n'y a pas ici de fonction objective à optimiser mais un coût minimal à respecter.
Affectation
Ce problème consiste à affecter des tâches à des ouvriers de manière bijective en respectant un certain coût. L'instance 1 comporte 16 variables avec comme domaine $\{0, 1\}$ ainsi que 9 contraintes. L'instance 2 comprend 25 variables et 11 contraintes.
Ordo
Problème d'ordonnancement avec 15 variables et 29 contraintes ; les domaines sont de taille 1000.

4.3 Résultats expérimentaux

Les valeurs fournies dans les tableaux correspondent à la valeur arrondie d'une moyenne effectuée sur 15 esssais pour chaque test. Nous mesurons le nombre total d'individus générés dans la population de couverture ainsi que le nombre total de mouvements de recherche locale effectués sur la population d'intensification. Pour la recherche locale seule, nous indiquons également le taux de succès. L'indication "-" signale qu'aucune réponse n'a été obtenue après 1 heure de calcul. Les réglages sont identiques pour les paramètres communs.
Nous comparons d'abord le nombre d'individus générés dans la population de couverture avec ACL et ACL+ACG pour obtenir la première solution. Dans le cas de ACL seul, ce nombre correspond au nombre de noeuds d'un arbre de backtracking classique. Le tableau 1 met en évidence l'efficacité de la phase d'intensification pour construire une première solution par rapport à un algorithme de backtracking traditionnel. Cet avantage est fonction de la structure du problème et de son nombre de solutions. Sur l'exemple *S+M=M*, qui n'a qu'une seule solution, la phase d'intensification a plus de mal à trouver cette solution que pour des problèmes possédant beaucoup de solutions comme *Ordo*.
Le tableau 1 met également en avant l'efficacité de la phase d'adaptation à la consistance locale pour guider la recherche locale (comparaison ACL+ACG vs. ACG seul), en particulier sur le problème *S+M=M*. Lorsque les problèmes

Nombre de solutions	Nb ind. gén.	Nb mvts
3	32	72
5	34	74
7	80	135

TABLE 2 – Obtention des solutions (ACL+ACG) sur Affectation-2

sont faciles pour la recherche locale (i.e. beaucoup de solutions bien réparties), l'apport de la première phase est moins significatif (pour le problème *Sac* par exemple). Nous pouvons de plus souligner que, en particulier pour le problème d'affectation, la robustesse de ACG seul est mauvaise, c'est à dire que l'écart type sur les différents essais au niveau du nombre de mouvements nécessaires à l'obtention d'une solution est très grand contrairement à la combinaison ACL+ACG qui est très stable.
Enfin, nous évaluons le coût calculatoire pour l'obtention de plusieurs solutions par ACL+ACG sur le problème *Affectation-2*. Ce problème possède 19 solutions, que ACL seul trouve en 148 noeuds. Par rapport à ACG seul, cette possibilité d'obtenir des solutions différentes réparties sur l'espace de recherche est un atout certain. Le tableau 2 souligne l'efficacité de la combinaison des mécanismes qui progressent conjointement pour obtenir rapidement un ensemble de solutions différentes. Ceci peut s'avérer particulièrement intéressant dans un certain nombre de problèmes.

5 Conclusion

Nous avons présenté un modèle général évolutionniste pour la résolution des CSP. Ce modèle repose sur une collaboration entre des méthodes de propagation de contraintes et de recherche locale. L'originalité de cette approche réside dans la représentation uniforme de l'espace de recherche par des populations devant s'adapter à des critères de consistance différents. Ceci permet à la fois une couverture exhaustive de l'ensemble des solutions ainsi que l'emploi de méthodes performantes pour intensifier la recherche de ces solutions. Nous avons montré sur quelques exemples que la combinaison de méthodes complètes et incomplètes, facilitée par notre modèle, permettait d'obtenir de bons résultats par rapport à une utilisation indépendante de ces techniques. De plus, ce concept évolutionniste unifié permet d'envisager diverses extensions et voies de recherche évoquées section 3.6.

Références

[1] K.R. Apt. The rough guide to constraint propagation. In Springer-Verlag, editor, *Proc. of the 5th International Conference on Principles and Practice*

of Constraint Programming (CP'99), volume 1713 of LNCS, pages 1–23, Springer-Verlag, 1999. (Invited lecture).

[2] F. Benhamou. Heterogeneous constraint solving. In M. Hanus and M. Rodriguez Artalejo, *Proceedings of the Fifth International Conference on Algebraic and Logic Programming, ALP'96*, volume 1139 of LNCS, Springer-Verlag, 1996.

[3] F. Focacci, F. Laburthe and A. Lodi. Local search and constraint programming. In Fred Glover and Gary Kochenberger, *Handbook of Metaheuristics*, Kluwer, 2002.

[4] F. Glover and M. Laguna. Tabu search. Kluwer Academic Publisher, 1997.

[5] D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. Reading, MA :Addison-Wesley, 1989.

[6] J.K. Hao and P. Galinier. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4) :379–397, 1999.

[7] J.H. Holland. Adaptation in Natural and Artificial Systemes. University of Michigan Press, 1975.

[8] N. Jussien and O. Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1) :21–45, 2002.

[9] N. Jussien and O. Lhomme. Vers une unification des algorithmes de résolution de csp. Dans les actes des *8ièmes Journées nationales sur la résolution pratique de problèmes NP-complets (JNPC'02)*, pages 155–168, 2002.

[10] S. Kirkpatrick, C. Gelatt and M. Vecchi. Optimization by simulated annealing : an experimental evaluation. *Science*, (220) :671–680, 1983.

[11] A. Mackworth. Encyclopedia on artificial intelligence, chapter constraint satisfaction. J. Wiley, 1987.

[12] K. Mariott and P.J. Stuckey. Programming with constraints, an introduction. MIT Press, 1998.

[13] R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28 :225–233, 1986.

[14] V. Tam and P.J. Stuckey. Improving evolutionary algorithms for efficient constraint satisfaction. *The International Journal on Artificial Intelligence Tools*, 2(8), 1999.

[15] E. Tsang. Foundations of constraint satisfaction. Academic Press, London, 1993.

[16] M. Vasquez and A. Dupont. Filtrage par arc-consistance et recherche tabou pour l'allocation de fréquence avec polarisation. Dans les actes des *8ièmes Journées nationales sur la résolution pratique de problèmes NP-complets (JNPC'02)*, 2002.

[17] D.L. Waltz. The psychology of computer vision, chapter generating semantic descriptions from drawings of scenes with shadows. Mc Graw Hill, 1975.