7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

COLONIES DE FOURMIS POUR L'ALLOCATION DE LA FIABILITE : SYSTEMES SERIE-PARALLELE

F.BELMECHERI, A.YALAOUI, E.CHÂTELET

Université de Technologie de Troyes
Institut Charles Delaunay
Optimisation des Systèmes Industriels
ICD-OSI
12, rue Marie Curie, BP 2060
10010 Troyes cedex
farah.belmecheri@utt.fr, alice.yalaoui@utt.fr, eric.chatelet@utt.fr

RESUME : *Nous présentons dans cet article un outil d'aide à la conception et à l'amélioration des systèmes, permettant d'atteindre des objectifs fixés en termes de fiabilité, et de coût. Cette étude porte sur l'allocation de la fiabilité d'un système simple série-parallèle, considéré comme non réparable. Ce problème d'optimisation de la fiabilité est NP-difficile. L'approche utilisée dans cette étude pour résoudre ce problème repose sur l'utilisation d'une technique méta heuristique ; celle des colonie de fourmis (Ant Colony otpimization ACO). Les résultats obtenus dépassent les performances des méthodes existantes.*

MOTS-CLES : *Optimisation de la fiabilité, système série-parallèle, colonies de fourmis*

INTRODUCTION

Dans l'industrie, les entreprises accordent une grande importance à la gestion de la qualité et des dysfonctionnements, que ce soit pour l'outil de production, ou pour les produits fabriqués. Ceci les amène à développer des politiques de gestion des coûts liés à la fiabilité des composants utilisés dans la production des produits fabriqués ou à la fiabilité des processus mis en œuvre. La gestion des coûts intervient dès la phase de conception du système considéré (produit finis ou outil de production, par exemple). Dans cette phase de conception, l'un des indicateurs primordiaux à prendre en compte (surtout pour les systèmes de sécurité, les produits dont l'usage peut présenter des risques, etc..), qui mesure les performances du système et de ses composants, est la fiabilité.

Un système est dit performant s'il répond aux besoins de son utilisateur et au moindre coût. Cela signifie que la conception ne se limite pas seulement à répondre au besoin exprimé en termes de fonctionnalités, mais aussi à considérer les deux critères importants que sont la fiabilité et le coût. La prise en compte de ces deux critères permet de garantir le niveau de performances des systèmes considérés.

De nombreux auteurs (A. Tzafesta, 1980), (DW. Coit et A. E. Smith, 1996), (A. Yalaoui *et al.*, 2004a), (A. Yalaoui *et al.*, 2004b) ont proposé des classifications qui sont faites en fonction de plusieurs critères, comme le type de système étudié (réparable,non réparable,série,parallèle) par exemple.

(N. Nahas et M. Nourelfath , 2005) ont développé un algorithme basé sur un système de fourmis (*Ant System* : AS) pour un problème d'optimisation de la fiabilité sur des systèmes de type série. Les auteurs ont étudié ce système en maximisant sa fiabilité sous contraintes du coût. Ils ont considéré un système de plusieurs composants mis en série. Pour chaque composant, il existe différentes technologies disponibles avec des coûts, des poids, et des fiabilités différentes. Le problème de conception est de choisir la meilleure combinaison des technologies pour maximiser la fiabilité avec un coût donné.

(Y. C. Liang et A. E. Smith, 2004) proposent l'ACO (*Ant Colony optimzation*) pour des problèmes d'allocation de redondance (RAP : *Redundancy Allocation Problem*) sur des systèmes parallèle-série, en maximisant la fiabilité du système. (R. Meziane *et al.*, 2005) ont adopté l'approche UMGF *(Universal Moment generating Function*) avec l'algorithme ACA (*Ant Colony Algorithm*) pour maximiser la fiabilité sur ce type de systèmes .

(N. Nahas *et al.*, 2007) ont présenté une méta-heuristique ACO (*Ant Colony Optimization*) pour le problème d'allocation de redondance et de fiabilité dans les systèmes parallèle-série, en maximisant la fiabilité du système. Leur étude consiste en premier à choisir d'une manière probabiliste dans chaque sous-système de type parallèle

un nombre de composants en redondance. Ensuite, une fois le nombre de composants choisi, ils affectent d'une manière toujours probabiliste parmi les technologies disponibles, une technologie à chaque composant. À chaque choix probabiliste, l'ACO vérifie si le chemin choisi respecte les contraintes coûts.

(J. H. Zhao *et al.*, 2007) se sont inspirés des travaux de (Y. C. Liang et A. E. Smith, 2004). Ils ont utilisé l'algorithme ACS (*Ant System Colony*) pour résoudre le problème d'allocation combinatoire sur des systèmes parallèle-série, en maximisant la fiabilité du système. Les auteurs ont comparé leurs résultats avec ceux obtenus à partir des algorithmes ACO - RAP et GA – RAP (respectivement *Ant Colony Optimization - Redundancy Allocation Problem* et *Genetic Algorithm - Redundancy Allocation Problem*). Les résultats s'avèrent meilleurs que ceux trouvés avec l'ACO et l'AG. Ils ont trouvé une meilleure fiabilité avec un coût minimum en moins d'itérations.

Les systèmes série-parallèle n'ont pas été beaucoup étudiés. On trouve en particulier ceux de (A. Yalaoui *et al.*, 2004b). Ces auteurs ont développé une méthode heuristique qui est l'algorithme YCC-SP. L'objectif est de minimiser le coût du système sous contrainte de fiabilité. La méthode YCC-SP a des performances limitées car elle utilise la programmation dynamique. Afin d'améliorer les performances de cette approche, il est intéressant d'utiliser une autre méthode de résolution.

Nous nous sommes donc intéressés à l'étude de l'optimisation de la fiabilité dans ces systèmes avec l'ACO. Notons qu'aucune méthode ACO n'a été appliquée sur ce type de système. Notre objectif est la minimisation du coût du système sous contraintes de fiabilité en nous inspirant de l'approche YCC-SP.

Cet article est divisé en cinq parties. La première partie présente brièvement les principes et formules des colonies de fourmis. La deuxième partie passe en revus les différentes hypothèses de travail qui permettent de modéliser le système à étudier. Cette partie présente aussi un rappel de l'algorithme YCC-SP (A. Yalaoui *et al.*, 2004b). La troisième partie est consacrée à l'application des colonies de fourmis au système étudié. La quatrième partie présente les résultats obtenus avec l'algorithme colonies de fourmis (ACO), la méthode YCC-SP et la méthode d'énumération complète. La cinquième et dernière partie présente une conclusion sur les résultats obtenus et les différentes perspectives.

1. PRINCIPES ET FORMULES DES COLONIES DE FOURMIS

Les algorithmes dits de colonies de fourmis forment une classe de méta-heuristiques proposée pour les problèmes d'optimisation difficile discrets.

(M. Dorigo *et al.*, 1996) sont les premiers à avoir proposé l'algorithme d'optimisation par les colonies de fourmis (ACO) pour le problème du voyageur de commerce (*Traveling Salesman Problem* TSP) Il se sont inspiré du comportement collectif des fourmis qui ont pour objectif de trouver le plus court chemin de leur nid à la source de nourriture.

D'après (J. Dréo *et al.*, 2005) l'algorithme de base des colonies de fourmis « *Ant System* » (A. Colorni *et al.*, 1992) pour le problème du voyageur de commerce se déroule de la manière suivante :

Soit une population de m fourmis et un graphe constitué de n villes à visiter. A chaque itération t ($1 \leq t \leq t_{max}$), chaque fourmi k (k=1,…, m) construit un parcourt complet permettant de visiter les n villes. Une fois à la ville i, la fourmi va choisir la ville j qu'elle va visiter. Ensuite déterminer la liste J_i^k des villes non visitées, qui définit les mouvements possibles quand la fourmi k est sur la ville i. Ce choix de la ville se fait en partie en considérant aussi l'inverse de la distance entre les villes : $\eta_{ij}=1/d_{ij}$, appelée *visibilité*. Cette information heuristique est utilisée pour diriger le choix des fourmis vers des villes proches, et éviter les villes trop lointaines. La quantité de phéromone $\tau_{ij}(t)$ déposée sur l'arête reliant les deux villes (i, j), appelée *intensité de la piste* guide les fourmis. Ce paramètre définit l'attractivité du trajet entre i et j et change à chaque passage d'une fourmi. Avec ces différentes informations, on définit les probabilités associées aux choix des différents chemins qui sont données par la formule suivante :

$$P_{ij}^k = \begin{cases} \dfrac{(\tau_{ij}(t))^\alpha \times (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha \times (\eta_{il})^\beta} & si \quad j \in J_i^k \\ 0 & si \quad j \notin J_i^k \end{cases} \qquad (1)$$

où α et β sont deux paramètres contrôlant l'importance de l'*intensité de la piste* $\tau_{ij}(t)$, et de l'information heuristique $\eta_{ij.}$. Une fois que la fourmi a construit un parcourt complet, chacune d'elles laisse une certaine quantité de phéromone $\Delta\ \tau_{ij}(t)$ sur le chemin entre la ville i et la ville j (formules (2) et (3)). C'est une quantité qui dépend de la *qualité* de la solution trouvée.

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \qquad (2)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & si \ \ (i,j) \in T^k(t) \\ 0 & si \ \ (i,j) \notin T^k(t) \end{cases} \qquad (3)$$

où $T^k(t)$ est le trajet effectué par la fourmi k à l'itération t, la longueur $L^k(t)$ de la tournée et Q est un paramètre fixe de la quantité de phéromone déposée.

Pour éviter d'avoir des solutions sous optimales (optimum local), il existe un processus d'évaporation qui permet de pénaliser les solutions non faisables. Cela signifie qu'à chaque itération, une piste donnant des solutions non fai-

sables aura une grande chance de ne pas être empruntée à la prochaine itération.

La règle de mise à jour des pistes est la suivante :

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (4)$$

Où ρ est le taux d'évaporation.
Le résumé de l'algorithme AS pour le problème du voyageur de commerce est comme suit :

- Pour t=1,.., t_{max}
  - Pour chaque fourmi k=1,…,m
    - Choisir une ville au hasard de départ parmi les n
    - Pour chaque ville non visitée i
      - Choisir une ville j dans la liste J_i^k des villes restantes, selon la formule (1)
  - Mise à jour des phéromones (3), évaporer les pistes selon la formule (4)
- Fin pour.

2. PROBLEMATIQUE

2.1 Hypothèses

Le système considéré est constitué de sous systèmes mis en parallèle. Chaque sous-système est quant à lui constitué d'un ensemble de blocs reliés en série. Chaque bloc peut représenter alors une machine, un composant ou encore une action en fonction du système modélisé. Pour chaque bloc, on dispose sur le marché de plusieurs articles qui sont aptes à remplir la mission du bloc. Chaque article est caractérisé entre autres par sa fiabilité et son coût. L'objectif est de trouver l'option à affecter à chaque bloc en respectant les objectifs en terme de coût et de fiabilité du système dans son ensemble. Les différentes hypothèses retenues pour l'étude de ce système sont les suivantes :

- Les composants du système étudié sont non réparables (où s'ils le sont, on s'intéresse à leurs fiabilités à la fin d'un horizon donné) et indépendants, ce qui signifie que la défaillance d'un composant n'influe pas sur les autres composants du système.
- Les composants et le système n'ont que deux états possibles marche ou panne.

2.2 Notations

Le système série-parallèle (Figure 1) est composé de K sous-systèmes en redondance active. Chaque sous-système i (i=1,…K) est constitué de composants en série. Les variables de décision du problème sont les fiabilités r_{ij} des composants (i=1,.., K et j=1,.., n_i). Nous supposons que la fiabilité des composants ne peut prendre qu'un nombre fini m_{ij}, correspondant aux différentes marques de composants disponibles sur le marché pour une même fonction. Les valeurs de ces fiabilités sont regroupées dans l'ensemble $\Omega_{ij} \in \{p_{ij1}, p_{ij2},.., p_{ijmij}\}$ (i=1,.., K et j=1,…, n_i) de telle sorte que $p_{ijt}<p_{ijl}$ pour $t < l$. On note c_{ij} le coût associé à r_{ij}, avec f_{ij} l'application telle que $c_{ij}=f(r_{ij})$.

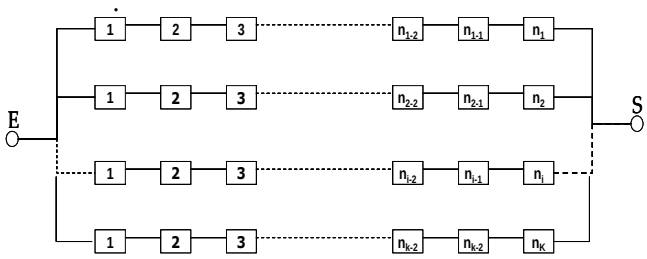

Figure 1.Système série-parallèle

2.3 Modélisation

(A. Yalaoui *et al.*, 2004b) ont proposé plusieurs modèles pour ce problème. Nous nous basons sur l'un d'entre eux que nous rappelons ci-dessous. Ce modèle est basé sur la définition d'une nouvelle variable de décision notée x_{ijl}. Les autres notations utilisées sont :

R_{min}	L'objectif de fiabilité minimale pour le système.
K	Le nombre de sous-systèmes.
n_i	Le nombre de composants dans le sous-système i (i=1,.., K).
r_{ij}	La fiabilité du composant j du sous système i (j=1,.., n_i), (i=1,.., K).
m_{ij}	Le nombre d'articles disponibles pour le composant j du sous-système i (j=1,.., n_i), (i=1,.., K).
c_{ijl}	Le coût de l'article l du composant j du sous-système i (l=1,.., m_{ij}), (j=1,.., n_i), (i=1,.., K).
p_{ijl}	La fiabilité de l'article l du composant j du sous système i (l=1,.., m_{ij}), (j=1,.., n_i), (i=1,.., K).
x_{ijl}	(i=1,…k, j=1,… n_i, et l=1,… m_{ij}) est égale à 1 si le composant de la fiabilité p_{ijl} est utilisé, soit égale à 0 sinon.

$$Min \quad \sum_{i=1}^{k}\sum_{j=1}^{n_i}\sum_{l=1}^{m_{ij}} C_{ijl}\, x_{ijl} \quad (5)$$

$$1-\prod_{i=1}^{k}\left(1-\prod_{j=1}^{n_i}\prod_{l=1}^{m_{ij}}\left(p_{ijl}\right)^{x_{ijl}}\right) \geq R_{\min} \quad (6)$$

$$\sum_{l=1}^{m_{ij}} x_{ijl} = 1, i = \{1,2,..k\}, j = \{1,2,..n_i\}, l = \{1,2,..m_{ij}\} \quad (7)$$

$$x_{ijl} \in \{0,1\} \quad (8)$$

(A.Yalaoui *et al.*, 2004b)

On notera que la contrainte (7) garantit l'affectation d'un et un seul article à chaque composant du système.

2.4 Méthode YCC-SP

Les travaux présentés dans cet article s'appuient sur la méthode YCC-SP. (A. Yalaoui *et al.,* 2004b) ont proposé une approche basée sur la programmation dynamique, reposant sur une analogie entre l'allocation de fiabilité et le problème de sac à dos. Cette approche peut être résumée dans la Figure 2, avec R_i la fiabilité du sous-système i (i=1,.., K) et R la fiabilité du système.

Pour chaque sous système $i=1,\ldots K$, calculer les bornes de fiabilité $\underline{R_i}$ et $\overline{R_i}$ (formule (9)).

a. Rechercher les N_i solutions réalisables telles que $\underline{R_i} \le R_i \le \overline{R_i}$ par la programmation dynamique.

b. Résoudre le système global avec la programmation dynamique en utilisant comme contrainte la fiabilité du système $R \ge R_{\min}$. Garder la solution de plus petit coût C^* en respectant cette contrainte.

Figure 2. Algorithme d' YCC-SP

$$\overline{R_i} = \prod_{j=1}^{n_i} p_{ijm_{ij}} \qquad \underline{R_i} = \max\left\{1 - \frac{1-R_{\min}}{\prod_{q=1,q\neq i}^{k}(1-\overline{R_q})}; \prod_{j=1}^{n_i} p_{ij1}\right\} \quad (9)$$

En effet, cet algorithme YCC-SP est basé sur des changements de variables. Ces derniers permettent de faire des analogies entre le problème d'allocation et le problème de sac à dos et d'utiliser la programmation dynamique (A. Yalaoui *et al.*, 2004b). Ces changements de variables génèrent une perte de précision dans les calculs. Cette méthode de complexité pseudo-polynomiale converge vers l'optimum. Mais cette convergence ne garantis pas l'optimum en particulier pour des problèmes de grande taille (quand le nombre d'articles total dans le système est supérieur à 50).

Nous allons garder dans notre nouvelle méthode de résolution les 2 étapes de YCC-SP, mais nous proposons de remplacer la programmation dynamique par les colonies de fourmis.

3. COLONIES DE FOURMIS POUR LE PROBLEME D'OPTIMISATION DE LA FIABILITE

La démarche utilisée pour résoudre l'allocation de fiabilité dans le système série-parallèle se décompose en 2 étapes, suivant la méthode YCC-SP. La première étape résout le problème relatif aux sous-systèmes, où l'on recherche dans YCC-SP l'ensemble des solutions telles que $\underline{R_i} \le R_i \le \overline{R_i}$.

La deuxième étape résout le problème global. L'algorithme ACO-SS (*Ant Colony Optimisation*– Sous Systèmes) est appliqué sur les sous-systèmes, et l'algorithme ACO-SG (*Ant Colony Optimisation*–Système Global) (Figure 6) est appliqué sur le système global. Nous allons donc présenter ces deux algorithmes.

3.1 Décomposition du problème en étapes (n_i étapes)

Cette partie présente l'application du fonctionnement général des colonies de fourmis présenté dans la partie 1 à la résolution du problème pour le sous-système i (i=1,.., K).

L'objectif est donc de trouver toutes les configurations, pour chaque sous-système i (i=1,.., K) qui ont une fiabilité R_i comprise entre $\underline{R_i}$ et $\overline{R_i}$.

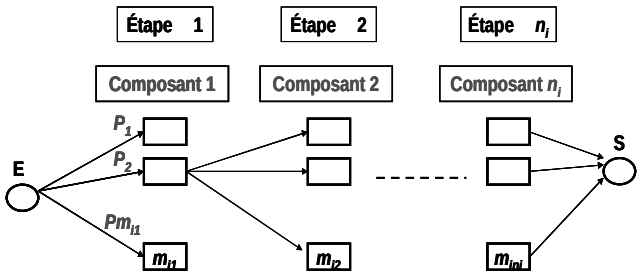

Figure 3. Application d'ACO-SS dans un sous-système i

Pour chaque sous-système i (i=1,.., K), considérons une population de m fourmis. Chaque fourmi k (k=1,.., m) va construire une solution pour l'allocation de fiabilité du sous-système i. Cette construction se fait en n_i étapes (Figure **3**). A l'étape j (j=1,.., n_i), la fourmi choisis la ville visitée parmi les m_{ij} disponibles. Le choix de la ville correspond au choix de l'article l (l=1,.., m_{ij}).

3.2 Probabilités de transition

A chaque chemin est associé une probabilité P_{ijl} (formule (11)) de telle sorte que $\sum_{l=1}^{m_{ij}} P_{ijl} = 1$.

Ces probabilités sont calculées à partir des taux de phéromones $\tau_{ijl}(t)$ (formule (12)) et d'une information heuristique $\eta_{ijl.}$ (formule (10)) qui guide les fourmis.

$$\eta_{ijl} = \frac{p_{ijl}}{c_{ijl}} \quad (10)$$

$$P_{ijl} = \frac{(\tau_{ijl}(t))^{\alpha_{ss}} \times (\eta_{ijl}(t))^{\beta_{ss}}}{\sum_{l=1}^{m_{ij}} \left((\tau_{ijl}(t))^{\alpha_{ss}} \times (\eta_{ijl}(t))^{\beta_{ss}}\right)} \quad (11)$$

où α_{ss} et β_{ss} sont des paramètres qui permettent de moduler l'importance de l'information phéromone et de l'information heuristique.

Notons que $\tau_{ijl}(t)$ rend compte de l'intérêt de choisir l'article l (l=1,.., m_{ij}) pour le composant j (j=1,.., n_i) du sous-système i (i=1,.., K). Quand une fourmi se trouve à l'étape j-1 au niveau de la ville (l'article) l, on utilise la

simulation d'une réalisation d'une loi discrète. Celle-ci est effectuée avec l'ensemble des réalisations possibles (1,2,…, *l*,…, m_{ij}) et leurs probabilités associées (P_{ij1}, P_{ij2},…, P_{ijl},…, P_{ijmij}). Le principe général de cette simulation est de partir d'une réalisation *u* de la loi uniforme U (0,1) (qui permet de simuler une probabilité) et de chercher la réalisation γ

($1 \le \gamma \le m_{ij}$) telle que $\sum_{l=1}^{\gamma-1} P_{ijl} \le u \le \sum_{l=1}^{\gamma} P_{ijl}$.

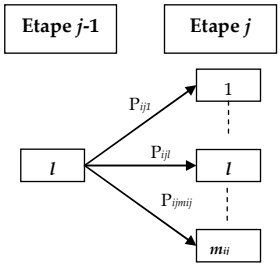

Figure 4. Le choix probabiliste d'un article *l* (*l*=1,.., m_{ij}) pour l'allouer au composant *j* (*j*=1,.., n_i).

3.3 Mise à jour des phéromones

Une fois que l'ensemble de la population des *m* fourmis à l'itération *t* a construit une solution, il y a une mise à jour des phéromones. Les formules (12), (13), (14) permettent d'augmenter le taux de phéromones τ_{ijl}.

Une fois que l'ensemble de la population des *m* fourmis, à l'itération *t* (*t*=1,.., t_{max}) a construit une solution, les phéromones sont mise à jour avec les formules (12), (13), (14). La nouvelle trace de phéromone $\tau_{ijl}(t)$ est le reste après évaporation (ρ_{ss} est la vitesse d'évaporation) de l'ancienne $\tau_{ijl}(t\text{-}1)$.

$$\tau_{ijl}(t) = \rho_{ss} \times \tau_{ijl}(t-1) + \sum_{f=1}^{m} \Delta\tau_{ijl}{}^{f} \qquad (12)$$

$$\Delta\tau_{ijl}^{k} = \begin{cases} Q_{ss} \times penalité_k \times (1/C_i^k) & \text{Si la fourmi } k \text{ a visité } (j,\ l) \\ 0 & \text{Sinon} \end{cases} \qquad (13)$$

$$penalité_k = \begin{cases} \left[C_i^* / C_i^k\right]^{Ass} & Si \quad \underline{R_i} \le R_i^k \le \overline{R_i} \\ \left[R_i^f / \underline{R_i}\right]^{Bss} & Si \quad R_i^k \le \underline{R_i} \end{cases} \qquad (14)$$

On rajoute au $\tau_{ijl}(t\text{-}1)$ une quantité de phéromone qui dépend de la qualité des solutions obtenues à l'itération *t*. La qualité de la solution résulte du choix de l'article *l* (*l*=1,.., m_{ij}) du composant *j* (*j*=1,.., n_i) du sous-système *i* (*i*=1,.., *K*).

L'augmentation de phéromone (formule (13)) est proportionnelle à l'inverse du coût de la solution (car on veut minimiser le coût), à un paramètre Q_{ss} qui module et pénalise les solutions. La pénalité (formule (14)) repose sur le fait que l'on veut favoriser toutes solution qui respectent la contrainte $\underline{R_i} \le R_i^k \le \overline{R_i}$ (où R_i^k représente la fiabilité du sous-système *i* obtenue par la fourmi *k)*. Pour ces solutions, plus le coût C_i^k (correspondant à R_i^k) est faible, plus l'augmentation des phéromones est importante. Notons que C_i^* représente le meilleur coût connu à l'itération *t*. Dans le cas où la fiabilité R_i^k est inférieure à $\underline{R_i}$, nous avons proposé d'augmenter aussi les taux de phéromones, de manière à éviter d'avoir un minimum local. Ces augmentations sont modulées par les paramètres A_{ss} et B_{ss} qui sont relatifs aux pénalités.

3.4 Algorithme ACO-SG

Notons N_i le nombre de solutions obtenues pour le sous-système *i* (*i*=1,.., *K*) telles que ces solutions réalisables dans chaque sous-système *i* (*i*=1,.., *K*) seront utilisées pour la résolution du système global.

La démarche utilisée pour la résolution du système global (Figure 5) est la même que pour les sous-systèmes. La seule différence concerne la mise à jour des phéromones.

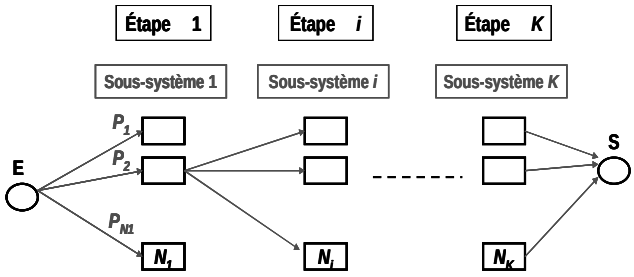

Figure 5. Application d'ACO-SG dans le système global

Chaque fourmi *k* (1=1,.., *m*) construit une solution pour l'allocation de fiabilité du système. A l'étape *i* (*i*=1,.., *K*), on choisit une solution *n* (*n*=1,.., N_i) obtenue par l'ACO-SS du sous-système *i* (*i*=1,.., *K*). Cette construction se fait avec des probabilités de transition P_{in} (formule 15) :

$$P_{in} = \frac{(\tau_{in}(t))^{\alpha} \times (\eta_{in}(t))^{\beta}}{\sum_{n=1}^{N_i} \left((\tau_{in}(t))^{\alpha} \times (\eta_{in}(t))^{\beta}\right)} \qquad (15)$$

Où P_{in} est la probabilité de choisir la solution *n* (*n*=1,.., N_i) du sous-système *i* (*i*=1,.., *K*) ; $\tau_{in}(t)$ est l'intensité de la trace de phéromone associée au choix de la solution *n* (*n*=1,.., N_i) à partir de l'étape *i*-1. L'information heuristique (formule (16)) aide la fourmi à choisir la solution *n* (*n*=1,.., N_i) du sous-système *i* (*i*=1,.., *K*) ayant une bonne fiabilité à moindre coût. α et β sont respectivement le paramètre d'intensité de phéromone et le paramètre d'information heuristique pour le problème du système global. Après construction des solutions, la mise à jour des phéromones utilise les formules (17), (18) et (19).

$$\eta_{in} = R_{in} / C_{in} \qquad (16)$$

$$\tau_{in}(t) = \rho \times \tau_{in}(t-1) + \sum_{k=1}^{m} \Delta\tau_{in}^{k} \quad (17)$$

Où ρ est toujours la vitesse d'évaporation. Pour la mise à jour, on utilisera aussi le paramètre Q qui permet de moduler et pénaliser les chemins.

$$\Delta\tau_{in}{}^{k}(t) = \begin{cases} Q \times \textit{pénalité}_k \times (1/C^k) & \text{Si la fourmi } k \text{ a visité } (i,n) \\ 0 \quad \text{Sinon} \end{cases} \quad (18)$$

$$\textit{penalité}_k = \begin{cases} \left[C^*/C^k\right]^A & Si \quad R^k \geq R_{\min} \\ \left[R^k/R_{\min}\right]^B & Si \quad R^k \leq R_{\min} \end{cases} \quad (19)$$

Si la solution trouvée par la fourmi k est réalisable ($R^k \geq R_{min}$), le taux de phéromone correspondant au choix n est augmenté avec le paramètre A. Plus le coût est petit, plus ce chemin sera favorable pour un nouvel emprunt. Si la solution n'est pas réalisable, on augmentera toujours le taux de phéromone avec le paramètre B, mais il reste que sa valeur est moins importante que celle de A. Ce paramètre B sert à éviter l'optimum local. La Figure 6 représente le pseudo-algorithme de l'ACO-SG. Dans le développement de cet algorithme, nous avons utilisé un critère d'arrêt. L'ACO-SS est arrêté avant le nombre total d'itérations, si on retrouve la même solution optimale (le plus petit coût) au bout d'un certain nombre d'itérations fixé au départ.

- Début
- Tester si le problème est réalisable
- Si oui continuer, sinon Fin
- Utiliser l'algorithme ACO-SS pour obtenir pour chaque sous-système i (i=1,.., K) les N_i solutions réalisables, représentées par leur fiabilité et leur coût.
- Boucle sur les itérations (itérations sur les N_i)
  - Boucle sur les fourmis
    - Boucle sur les sous-systèmes
      - Calculer P_{in} ($1 \leq n \leq N_i$)
      - Choix aléatoire de n
    - Fin de la boucle sous-systèmes
  - Fin de la boucle fourmis
  - Mise à jour de la meilleure solution obtenue jusqu'à présent

  Mise à jour des phéromones
  - Boucle sur les sous-systèmes
    - Boucle sur les fourmis
      - Mise à jour
    - Fin de la boucle fourmis
  - Fin de la boucle sous-systèmes
- Fin de la boucle itérations.
- Fin.

Figure 6. Algorithme ACO-SG appliqué au système

4. RESULTATS

Dans cette partie, les résultats obtenus avec notre approche par les colonies de fourmis sont présentés pour le problème d'allocation de la fiabilité. Avant, la démarche adoptée pour trouver les meilleurs réglages afin d'atteindre l'optimum est exposée. Les algorithmes ont été programmés avec le langage C++ 6.0, sur un Pentium 4 de RAM 512 Mo. Enfin, nous comparons ces résultats obtenus avec ceux obtenus par les méthodes YCC-SP, et avec la solution optimale obtenue par l'énumération complète.

4.1. Réglages des paramètres

Nous avons effectué une compagne de tests afin de trouver le meilleur réglage pour le système. Pour cela, nous avons généré 50 problèmes pour des tailles K=3 et K=4. A chaque sous-système, le nombre de composants n_i est généré aléatoirement de 1 à 7 de même que le nombre d'articles m_{ij} pour chaque composant.

Nous avons choisis d'étudier des systèmes à 3 et 4 sous-systèmes car cela se rapproche de cas réels typiques. On peut avoir 3 à 4 technologies mises en parallèle pour atteindre le niveau de fiabilité requis. L'objectif est d'affecter les bons composants à chaque technologie (K=3, 4) pour un meilleur fonctionnement et à moindre coût.

4.1.1. Réglages des paramètres des problèmes relatifs aux sous-systèmes

4.1.1.1 Présentation du protocole

Afin de proposer le meilleur réglage pour obtenir les N_i solutions des sous-systèmes i (i=1,.., K), nous avons généré 20 sous-systèmes de chaque taille. La taille correspond au nombre de composants n_i (n_i=*1,.., 7*). Cela veut dire que nous avons étudié d'abord 20 sous-systèmes qui ont 1 composant, puis une autre vingtaine de sous-systèmes dont le nombre de composant est 2, et ainsi de suite jusqu'aux sous-systèmes possédant 7 composants. On notera toujours que le nombre d'articles m_{ij} pour chaque composant est généré aléatoirement de 1 à 7.

Les réglages des paramètres pour les sous-systèmes ont été faits de manière à maximiser le nombre de solutions (N_i) associé à chaque sous-système. A chaque taille d'un sous-système, nous avons fait varier la valeur de α_{ss} et nous avons fixé les valeurs de β_{ss}, ρ_{ss}, Q_{ss}, A_{ss}, et B_{ss}. Puis on a gardé la valeur de α_{ss} qui donne le maximum de nombre de solutions (N_i) associé au sous-système i. On a refait la même opération en fixant la meilleure valeur trouvée de α_{ss} avec les paramètres fixes ρ_{ss}, Q_{ss}, A_{ss}, et B_{ss} et on fait varier β_{ss}. La même démarche se répète pour tous les paramètres et à toutes les tailles n_i (n_i=*1,.., 7*) des sous-systèmes. Les meilleures valeurs retenues sont résumées dans la partie 4.1.1.3.

4.1.1.2 Illustration des réglages des paramètres (problèmes relatifs aux sous-systèmes)

Nous présentons deux courbes (Figure 7) qui représentent l'amélioration du nombre de solutions de deux sous-problèmes relatifs aux sous-systèmes *x* et *y* en fonction de plusieurs valeurs du paramètre par exemple α_{ss}.

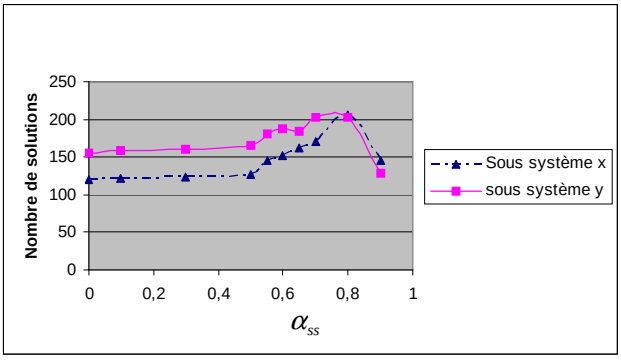

Figure 7. Améliorations des nombres de solutions dans les sous-systèmes *x* et *y* avec le paramètre α_{ss}.

Nous remarquons bien que pour les sous-systèmes *x* et *y* de même taille n_i= 7, la meilleure valeur de α_{ss} pour obtenir un maximum de solutions N_i est 0,8.

Nous avons remarqué que les réglages des paramètres des sous-problèmes dépendent du nombre total d'articles m_{ij}.

4.1.1.3 Présentation des résultats des problèmes relatifs aux sous-systèmes

Cette partie présente les résultats obtenus après les réglages des paramètres pour l'étude relative aux sous-systèmes. Ces derniers sont classés par différents nombres totaux d'articles m_{ij} disponibles.

	α_{ss}	β_{ss}	ρ_{ss}	Q_{ss}	A_{ss}	B_{ss}
1 et 2 composants (articles 1 à 10)	0,0009	0,0009	0,95	0,1	1	0,1
3 composants (articles 10 à 20)	0,55	0,005	0,95	0,01	1	0,1
4 Composants (articles 15 à 25)	0,55	0,01	0,99	0,1	1	0,1
5 et 6 composants (articles 25 à 35)	0,65	0,01	0,97	0,1	1	0,1
7 composants (articles 35 à 40)	0,8	0,01	0,95	0,1	1	0,1

Tableau 1. Les meilleures valeurs des paramètres pour différentes tailles des sous-systèmes

Plus le nombre total de composants augmente, plus les valeurs de α_{ss} et β_{ss} augmentent. De même, les paramètres ρ_{ss} et Q_{ss} qui varient selon le nombre de composants, mais plus sensiblement et de manière non monotone. Les paramètres A_{ss} et B_{ss} ne changent pas de valeurs car ils n'améliorent pas la qualité des résultats.

4.1.2. Réglages des paramètres du problème global

4.1.2.1 Présentation du protocole

Après avoir obtenus les N_i solutions de chaque sous-systèmes *i* (*i*=1,.., *K*), nous utiliserons ces derniers pour les réglages des paramètres du problème global. Pour le système avec *K*=3, nous avons adopté la même démarche que celle vue pour les sous-systèmes. Par exemple, nous avons fixé β=0,04, ρ=0,95, *Q*= 0,1, *A*=1, *B*=0,1 et avons fait varier α de 0,01 à 1. Nous avons retenus la valeur de α=0,04 car cette valeur donne le coût optimal ou parfois proche de l'optimum en moins d'itérations. Après, nous avons fixé cette valeur de α et gardé les mêmes valeurs de ρ, *Q*, *A*, *B*, puis nous avons fait varier β de 0,07 à 10. De la même façon, nous avons retenu la valeur qui donne l'optimum en moins d'itérations qui est 0,09. Nous avons donc gardé α=0,04, β=0,09 et nous avons fait varier ρ de 0,7 à 0,999. Puis nous avons retenus ρ=0,7 et fait varier *Q* de 0,1 à 10. Nous avons gardé la valeur de *Q* qui est 0,1. Nous avons remarqué qu'en faisant varier les paramètres *A* et *B* (en respectant la contrainte *A* > *B*), la qualité des résultats ne s'améliore pas. Nous avons donc retenus leurs valeurs qui sont respectivement 1 et 0,1.

Pour les systèmes avec *K*=4, nous avons procédé de la même manière mais avec des valeurs de paramètres différentes que celle vue pour *K*=3. Le Tableau 2 présente les meilleurs réglages pour *K*=3 et *K*=4.

	α	β	ρ	*Q*	*A*	*B*
***K*=3**	0,04	0,07	0,7	0,1	1	0,1
***K*=4**	0,04	0,09	0,95	0,1	1	0,1

Tableau 2. Les paramètres retenus pour les systèmes (*K*=3, 4)

4.1.2.2 Illustration des réglages du système global

Nous avons générés 50 problèmes (systèmes) de chaque taille (*K*=3, 4). L'objectif est d'obtenir les paramètres optimaux.

Définissons **E** comme étant l'écart moyen entre le nombre d'itérations obtenues pour les différents réglages et le nombre d'itérations obtenues pour un réglage donné. La Figure 8 représente l'évolution de E à chaque valeur du paramètre β de 0,09 à 5 pour les 50 systèmes à *K*=4 par exemple.

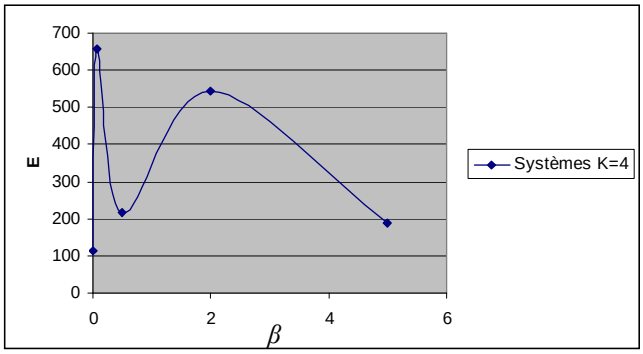

Figure 8. Evolution de E en fonction β pour *K*=4

Nous constatons que la valeur de E qui s'approche le plus du 0 est 115,3 correspondant à la valeur optimale β=0,09.

4.1.2.3 Présentation des résultats du système global

Une compagne de tests a été faite en générant 50 problèmes (systèmes) de chaque taille K=3 et K=4. Les tableaux 3 et 4 présentent les valeurs de E correspondantes aux paramètres α, β, et ρ. Notre objectif était d'obtenir les meilleures valeurs de ces paramètres. Nous nous sommes concentrés sur ces paramètres car ils influençaient sur la qualité des solutions contrairement à Q=0,1, A=1, et B=0,1 ($A \geq B$).

α =	**0,04**	**0,1**	**0,6**	**1**
E	**10,7**	29,8	NO	NO
β =	**0,07**	**2**	**5**	**10**
E	**10,9**	26,6	41,25	28
ρ =	**0,7**	**0,8**	**0,95**	**0,999**
E	**60,7**	67,44	65,6	65,6

Tableau 3. Les écarts moyens (E) correspondant aux paramètres α, β, et ρ pour les systèmes avec K=3

Le **Tableau 3** présente l'évolution de E en fonction des paramètres α, β, et ρ, où NO indique que l'optimum n'est pas atteint pour plus de 50% des problèmes générés.

Les valeurs de E pour les paramètres α, β, et ρ qui sont les plus faibles sont respectivement 10,7, 10,9 et 60,7. Cela nous indique que les meilleures valeurs des paramètres sont α=0,04, β =0,07 et ρ=0,7.

Le **Tableau 4** présente les résultats correspondant à K=4. Les meilleures valeurs des paramètres sont α=0,04 et β =0,09.

α =	**0,001**	**0,02**	**0,04**	**0,1**	
E	1518,62	1648,66	**969,37**	1416,87	
β =	**0,09**	**0,07**	**0,5**	**2**	**5**
E	**115,3**	658,28	215,28	542,57	188,33

Tableau 4. Les écarts moyens (E) correspondant aux paramètres α et β pour les systèmes avec K=4

4.2. Performances de la méthode

4.2.1. Présentation du Protocole

La compagne de tests a comparé 50 problèmes générés (systèmes) de chaque taille K=3 et K=4. A chaque sous-système i (i=1,.., K), le nombre de composants n_i et le nombre d'articles m_{ij} sont générés aléatoirement de 1 à 7.

4.2.2. Présentation des résultats

Les solutions obtenues sur les problèmes étudiés ont été comparées avec les résultats obtenus par les méthodes YCC - SP et de l'énumération complète.

Certes, la méthode d'énumération garantit l'optimum à 100%, mais cette méthode est pénalisée par la taille des problèmes (K et m_{ij} total) et les temps d'exécutions sont très importants.

La méthode YCC-SP atteint l'optimum à 80% des problèmes dans un intervalle m_{ij}= [3,50]. Pour l'intervalle m_{ij}= [50,100], l'optimum est atteint à 50%. Nous avons obtenus ces optimums en faisant varier les pas de précision (paramètre important dans YCC-SP) afin d'assurer l'optimum. De plus, cette méthode a des temps de calculs importants.

Avec l'approche « colonies de fourmis » l'optimum est atteint en fonction du nombre total d'articles disponibles dans le système. Dans l'intervalle m_{ij}= [3,100], l'optimum est atteint à 90%. Par contre dans l'intervalle m_{ij}= [100,196], l'optimum est atteint à 75 %.

5. CONCLUSION

Nous nous sommes intéressés dans cet article aux systèmes série-parallèle. Nous avons pris en compte le cas où les fiabilités des composants sont à choisir parmi un ensemble de valeurs discrètes. En effet, cette hypothèse rend compte de la disponibilité des composant sur le marché. Pour notre problème, qui est de minimiser le coût du système sous contrainte de fiabilité, nous avons utilisé l'algorithme de colonies de fourmis. L'approche se fait en deux phases.

Nous avons testé L'algorithme ACO-SS sur plusieurs problèmes. Il permet d'obtenir la solution optimale dans une majorité des cas, dans des temps de calculs très faibles, comparativement à l'énumération complète.

Une perspective intéressante peut être une étude de ce même type de système en hybridant les colonies de fourmis avec une recherche locale. Une autre perspective intéressante serait de développer une autre approche pour la résolution de ce type de système non basée sur l'approche en 2 étapes (approche de la méthode YCC-SP).

L'étude par la suite de systèmes plus complexes (série-parallèle-série) avec différentes approches est également une perspective intéressante.

REFERENCES

Coit DW. , AE. Smith, Reliability optimisation of series parallel systems using a Genetic Algorithm, *IEEE Transactions on Reliability*, 1996, vol. 45, n° 2, June, pp 254-260.

Colorni A., M. Dorigo, V. Maniezzo, Distributed optimization by Ant Colonies. Elsevier Publishing, 1992, pp 134-142.

Dorigo M., V. Maniezzo, A. Colorni, The Ant: System Optimization by a colony of cooperating Agents. *IEEE Transactions on Reliability*, 1996, vol. 26, pp 29-42.

Dréo J., Pétrowski, P. Siarry, E. Taillard, Métaheuristique pour l'optimisation difficile. Eyrolles. ISBN: 2-212-11368-4.

Liang Y-C., AE. Smith, Ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability*, 2004, vol. 53, n°3, pp 417-23.

Meziane R.,Y. Massim, A. Zeblah, A. Ghoraf, R. Rahli, Reliability optimization using ant colony algorithm under performance and cost constraints, *Electric Power Systems Research*, 2005, vol. 76, n°1-3, pp 1-6.

Nahas N., M. Nourelfath, Ant system for reliability optimization of a series system with multiple choice and budget constraints, *Reliability Engineering end System Safety*, 2005, vol. 87, n°1, pp 1-12.

Nahas N. , M. Nourelfath, D. Ait-Kadi, Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series parallel systems, *Reliability Engineering end System Safety*, 2007, vol. 92, n° 2, pp 211-222.

Tzafesta SG., Optimisation of system reliability: A survey of problems and techniques, *International Journal System Science*, 1980, vol.11, n°4, pp 455-486.

Yalaoui A., E. Châtelet et C. Chu, Allocation de fiabilité et de redondance : systèmes parallèle série, *Journal Européen des systèmes automatisés*, 2004a, vol. 38, n°1-2, pp 85-102.

Yalaoui A., C. Chu and E. Châtelet, Allocation de fiabilité dans les systèmes série parallèle, *MOSIM, 2004b*, pp 1097-1103.

Zhao J-H., Z. Liu, M-T. Daou, Reliability optimization using multiobjective ant colony system approaches, *Reliability Engineering and System Safety*, 2007, vol. 92, n°1, pp 109-120.