NABIL NAHAS

Optimisation de la performance de systèmes multi-composants assujettis à des défaillances aléatoires

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en génie mécanique
pour l'obtention du grade de Philosophiae Doctor (Ph.D.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2008

©Nabil Nahas, 2008

Résumé

Les travaux de cette thèse portent sur la conception optimale des systèmes de production constitués d'équipements ayant une fiabilité donnée. Les systèmes étudiés utilisent la redondance et/ou des stocks tampons comme technique d'amélioration de la performance. La méthodologie proposée développe des modèles de conception optimale, des méthodes d'évaluation et des algorithmes de résolution robustes et efficaces à base de métaheuristiques pour ces systèmes. Cette méthodologie est validée en l'appliquant à trois problèmes de conception optimale des systèmes de production caractérisés comme N-P difficiles :

1. Le problème d'allocation de la redondance (PAR) : Un algorithme hybride très efficace est proposé pour résoudre ce problème d'optimisation. Cet algorithme est basé sur une hybridation de l'algorithme à colonies de fourmis avec l'algorithme du grand déluge étendu. Les résultats numériques obtenus pour 33 problèmes tests, existant dans la littérature du PAR, montrent que l'algorithme proposé est très compétitif par rapport aux autres approches présentées dans la littérature.
2. Le problème d'allocation optimale des stocks : Deux types de lignes sont traitées, à savoir les lignes homogènes et les lignes non-homogènes. Une nouvelle approche basée sur l'algorithme du grand déluge étendu est proposée. Une étude comparative avec les meilleurs résultats publiés dans la littérature (recuit simulé) montre l'efficacité de cette approche tant au niveau du temps de convergence qu'au niveau de la qualité des solutions.
3. Le problème d'allocation optimale des stocks et de la redondance : Une méthode d'évaluation du taux de production d'une ligne de production série-parallèle est proposée et validée par simulation. Deux algorithmes utilisant des métaheuristiques, à savoir l'algorithme à colonies de fourmis et le recuit simulé, sont développés et comparés. Les résultats numériques obtenus, pour différentes instances générées aléatoirement, illustrent l'efficacité de l'algorithme à colonies de fourmis lorsqu'il est couplé à une procédure d'amélioration.

Avant propos

Je tiens tout d'abord à témoigner toute ma reconnaissance et ma profonde gratitude à mon directeur de thèse, le Professeur Daoud Ait-Kadi et à mon co-directeur de thèse, le Professeur Mustapha Nourelfath qui ont accepté de diriger cette thèse. Leurs nombreux conseils, leurs remarques scientifiques pertinentes m'ont guidé tout au long de ce travail et m'ont permis d'approfondir mes connaissances dans le domaine de génie industriel.

Par ailleurs, j'exprime toute ma reconnaissance envers les membres de jury de thèse qui m'ont accordé de leur temps et m'ont fait l'honneur de juger les travaux présentés dans cette thèse. Je pense notamment à M. Fayez Boctor Professeur à l'Université Laval ; M. Georges Abdul-Nour Professeur à l'Université du Québec à Trois-Rivières et M. Abdelhakim Artiba Professeur à l'Institut Supérieur de Mécanique de Paris (Supméca, Paris).

J'adresse tous mes remerciements à toutes les personnes que j'ai côtoyées durant ces trois années au sein du département de génie mécanique à l'Université Laval. Je pense plus particulièrement à toutes les personnes du CIRRELT et surtout à mes proches collègues Zouheir, Ayad, Souad et Redouane pour leur aide et leur bonne humeur. Je tiens à saluer mon collègue et ami Hakim avec qui j'ai eu l'occasion de collaborer dans quelques travaux de recherche et qui a montré une persévérance et un grand esprit d'analyse.

Enfin, j'aimerai offrir cette thèse à ma defunte mère qui nous a quittée avant l'achèvement de ce travail. Je remercie vivement mon père, mes frères et ma soeur pour leurs soutiens et encouragements pendant toutes ces années. Sans leur precieux soutien, cette thèse n'aurait jamais vu le jour.

Cette thèse, rédigée selon le principe d'insertion d'articles, est composée de trois articles cosignés avec Pr. Daoud Ait-Kadi et Pr. Mustapha Nourelfath. Pour chacun des articles présentés, j'ai agi à titre de chercheur principal. En tant que premier auteur, j'ai ainsi réalisé la conception des algorithmes, l'implantation informatique, la calibration

des parametres et l'analyse des résultats, ainsi que la rédaction de la première version de chacun des trois articles. Les professeurs Daoud Ait-Kadi et Mustapha Nourelfath ont révisé les articles, les modèles et les algorithmes proposés jusqu'à l'obtention des résultats finaux et la parution ou l'acceptation des articles.

Le premier article intitulé "Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series-parallel systems" cosigné avec Pr. Mustapha Nourelfath et Pr. Daoud Ait-Kadi, vient de paraître (février 2007) dans le journal "Reliability Engineering and System Safety".

Le second article intitulé "A new approach for buffer allocation in unreliable production lines" cosigné avec Pr. Daoud Ait-Kadi et Pr. Mustapha Nourelfath, a été publié en octobre 2006 dans le journal "International Journal of Production Economics".

Le troisième article intitulé "Selecting machines and buffers in unreliable series-parallel production lines" coécrit avec Pr. Mustapha Nourelfath et Pr. Daoud Ait-Kadi, a été accepté, en octobre 2007, dans le journal "International Journal of Production Research".

À ma defunte mère, à mon père et à mes frères et soeur

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction générale et revue de la littérature

1.1 Introduction

L'environnement industriel actuel est caractérisé par une concurrence vive dans la plupart des secteurs, par un marché fluctuant et par une technologie en progrès constant. Ce progrès implique des changements fréquents au niveau de la production (évolution des produits et des équipements), ainsi qu'une augmentation des coûts, surtout ceux des équipements, des produits semi-finis et du personnel. Pour s'adapter à cet environnement tout en demeurant compétitif, la préoccupation principale de chaque chef d'entreprise est de diminuer les coûts et augmenter la performance de son système de production. Les entreprises manufacturières doivent ainsi développer des moyens de production fiables et flexibles, maintenir un niveau élevé de la qualité des produits, optimiser et rationaliser l'utilisation des équipements et réduire les coûts de conception et d'exploitation de l'ensemble du système de production. Face à une telle demande croissante pour des systèmes très fiables, plus sécuritaires et moins chers et afin de garantir une continuité de service en cas de défaillance, plusieurs techniques sont généralement utilisées pour augmenter la performance (fiabilité, disponibilité, taux de production, etc.) de tels systèmes, par exemple :

1. augmenter leur niveau de redondance (allocation de la redondance),
2. introduire des stocks tampons entre les machines afin de les découpler les unes des autres,
3. augmenter la fiabilité des composantes du système (allocation de la fiabilité),
4. mettre en place des plans de maintenance corrective ou préventive.

Les travaux de cette thèse portent sur la conception optimale des systèmes de production constitués d'équipements ayant une fiabilité donnée. Les systèmes étudiés utilisent la redondance et/ou des stocks tampons comme technique d'amélioration de la performance. Nous nous intéressons au développement de modèles de conception optimale et d'algorithmes de résolution efficaces et robustes à base de méta-heuristiques pour ces systèmes. Dans ce chapitre, nous présenterons la problématique, les objectifs de notre travail et une revue de la littérature, ainsi qu'un résumé de la méthodologie poursuivie dans chacun des trois articles qui constituent cette thèse de doctorat.

1.2 Problématique et objectifs

Les travaux de cette thèse s'inscrivent dans une problématique générale de conception optimale de systèmes de production, en tenant compte du fait que les processeurs (machines) utilisés sont assujettis à des défaillances aléatoires. À chaque produit, on associe un processus qui établit la séquence des opérations à effectuer. Les opérations sont réalisées par des processeurs dont les caractéristiques opératoires se dégradent à l'usage. Commençons par illustrer l'importance des stocks tampons et de la redondance dans l'amélioration de la performance d'un système de production.

1.2.1 Lignes de production, stocks tampons et redondance

Les systèmes de production manufacturiers sont souvent constitués de machines séparées par des stocks intermédiaires. L'ensemble ainsi constitué est appelé dans la littérature ligne de production ou ligne de transfert, ou encore ligne de fabrication [35][27][1][39]. Dans cette thèse, nous utilisons principalement le terme ligne de production manufacturière, ou tout simplement ligne de production.

Lignes de production séries sans stocks tampons

Considérons des systèmes constitués de n machines opérant en série sans stocks tampons (figure 1.1). La défaillance d'une machine entraîne l'arrêt complet de la ligne de production. Ceci affecte négativement la disponibilité du système qui ne sera pas en mesure de produire de nouvelles pièces jusqu'à ce que la machine en panne soit réparée.

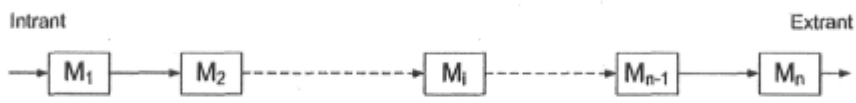

FIG. 1.1 – Ligne de production à n processeurs sans stocks tampons

Afin d'éviter de tels arrêts aléatoires et d'améliorer la disponibilité et le taux de production de la ligne, nous pouvons utiliser la redondance et/ou des stocks tampons. En effet, l'introduction des stocks tampons entre les machines permet de les découpler

les unes des autres, alors que grâce à la redondance, lorsqu'une machine tombe en panne, le flot de production est dirigé vers l'autre machine en parallèle.

Dans le cas où ces deux techniques sont utilisées, on parle alors de lignes de production séries-parallèles avec stocks tampons. La sous-section qui suit va illustrer comment ces techniques conduisent à une amélioration de la performance.

Lignes de production séries-parallèles avec stocks tampons

Nous considérons des lignes de productions composées de n processeurs et $(n-1)$ stocks tampons. Une portion de cette ligne est représentée à la figure 1.2. Ces configurations peuvent être rencontrées autant dans les systèmes manufacturiers, les chaînes logistiques que dans les entreprises de services. Chaque processeur i peut contenir k_i (avec $k_i \geq 1$) machines (notées m_{ij}, avec $j = 1$ à k_i) opérant en parallèle.

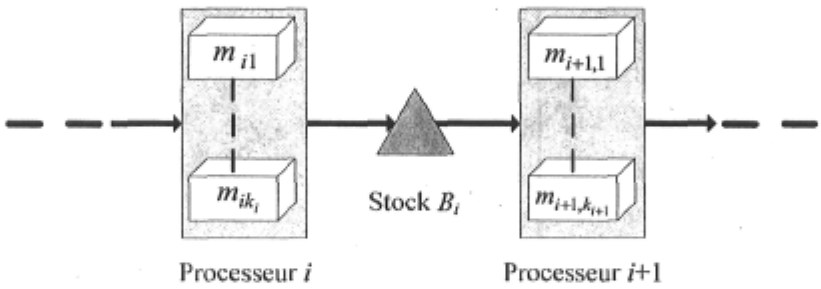

FIG. 1.2 – Portion d'une ligne de production série-parallèle avec stocks tampons

Le rôle de la redondance dans l'amélioration de la disponibilité et du taux de production de la ligne est évident puisqu'un processeur fonctionnant avec plusieurs machines redondantes ne sera considéré en panne que lorsque toutes les machines en parallèle sont en panne.

Afin de montrer le rôle des stocks, désignons par B_i le stock tampon entre les processeurs i et $(i+1)$. Si le processeur i tombe en panne (toutes les machines en parallèle sont en panne), le processeur $(i+1)$ poursuit le traitement des produits placés dans le stock B_i et ce jusqu'à ce que ce dernier se vide complètement. En l'absence de produits à traiter, le processeur $(i+1)$ s'arrête. On parlera dans ce cas d'arrêt par famine. Le processeur $(i+1)$ étant toujours hors d'usage, le processeur i continuera de produire tant et aussi longtemps que le stock B_i n'est pas plein. Dans le cas où B_i atteint sa capacité maximale, le processeur i cessera de produire. On parlera dans

ce cas d'un arrêt par blocage. On se limitera aux trois causes d'arrêt d'un processeur, soient : la défaillance accidentelle, le blocage ou la famine. On suppose que le ou les stocks d'alimentation et ceux des produits finis, ont des capacités infinies. Les stocks tampons sont généralement de capacité finie.

Les temps de traitement sont supposés connus. Ils peuvent être les mêmes (cas d'une ligne de production balancée ou homogène) ou peuvent différer d'un processeur à un autre (ligne non balancée ou non homogène). Dans le cas des lignes non balancées, des encours de production s'accumuleront entre les processeurs. Cette accumulation se produira également lorsque le processeur $(i+1)$ tombe en panne et le processeur i demeure en opération.

L'utilisation de plusieurs machines en parallèle ou un surdimensionnement des stocks engendreront des coûts d'installation et d'exploitation onéreux et possiblement inutiles. De même qu'un sous dimensionnement et un manque de machines redondantes pourraient mettre la capacité du système de production à répondre à la demande en péril.

Dans ce contexte, la performance du système sera évaluée par son taux de production et nous allons étudier les deux problèmes suivants :

1. Le problème d'allocation optimale des stocks.
2. Le problème d'allocation conjointe des stocks et de la redondance.

Par ailleurs, en absence des stocks tampons, le schéma parallèle-série correspond également à un diagramme de fiabilité. La performance du système peut ainsi être évaluée aussi par sa fiabilité. Il s'agira dans ce cas du problème d'allocation optimale de la redondance initialement étudié dans [32].

Ces trois problèmes sont d'une importance cruciale pour les systèmes de production. Avant de délimiter les objectifs spécifiques de notre étude, nous allons présenter ici la problématique sous-jacente à ces problèmes de conception optimale des systèmes de production.

1.2.2 Conception optimale des systèmes séries-parallèles utilisant des stocks tampons

Allocation optimale des stocks

Les stocks tampons, introduits entre les processeurs d'une ligne série pour améliorer la disponibilité, engendrent des coûts additionnels, et l'espace alloué est généralement limité. Si les stocks tampons sont judicieusement localisés et correctement dimensionnés, alors la défaillance d'un processeur n'entraîne pas nécessairement l'arrêt de la ligne de production. La problématique, consistant à trouver de quelle manière optimiser le déploiement des stocks tampons à travers un système de production, est très importante. L'objectif est de déterminer les quantités optimales de stocks à allouer à chaque zone tampon afin de maximiser l'efficacité de la ligne de production. Ce problème de conception est connu dans la littérature sous le nom du *problème d'allocation optimale des stocks*. Ce problème est N-P difficile [43]. Ceci implique que le temps de calcul nécessaire pour trouver une solution optimale risque d'augmenter exponentiellement avec la taille du problème, ce qui rend les méthodes exactes inefficaces face aux applications de taille importante.

Pour résoudre le problème d'allocation optimale des stocks, deux outils sont nécessaires :

- Une méthode d'évaluation de la performance de la ligne de production.
- Une méthode d'optimisation de l'allocation des stocks tampons qui utilise l'outil d'évaluation pour déterminer la fonction objectif à optimiser.

Deux objectifs découlent ainsi de cette problématique :

- Proposer une méthode d'évaluation de l'efficacité d'une ligne de production. Cette méthode doit être robuste et rapide. La robustesse implique une estimation de l'efficacité avec une grande précision. La rapidité signifie simplement que cette efficacité est calculée en un temps assez court pour permettre l'utilisation de la méthode dans l'algorithme d'optimisation.
- Développer un algorithme d'optimisation pour résoudre les problèmes de grandes tailles. Sachant que les méthodes exactes sont inefficaces face aux applications de taille importante, notre objectif ici est de concevoir et tester une nouvelle méthode approchée capable de fournir une solution très proche de l'optimale, et ce en un temps assez court.

Allocation optimale des stocks et de la redondance

La plupart des travaux publiés dans la littérature considèrent seulement le problème d'allocation des stocks tampons en considérant que les technologies des machines de la ligne de production sont déjà choisies, et que les quantités de stocks sont continues. La problématique abordée ici consiste à traiter un nouveau problème qui étend à la fois le problème d'allocation optimale des stocks et le problème d'allocation de la redondance consistant à sélectionner les machines. Dans notre approche, la conception optimale de la ligne de production consiste à faire une sélection conjointe des technologies des machines et de quantités discrètes de stocks tampons.

En se référant à la figure 1.2, le problème de conception de la ligne de production considéré consiste à déterminer, pour chaque processeur i, le nombre de machines d'une certaine technologie à placer en parallèle ainsi que les dimensions des stocks tampons. Nous considérons le problème pratique où :

- Pour chaque machine, plusieurs technologies sont disponibles dans le marché ; chacune de ces technologies étant caractérisée par un taux de production, une fiabilité, et un coût d'acquisition et d'exploitation.
- Les stocks tampons peuvent être choisis parmi une liste, supposée discrète, de versions possibles ; chacune de ces versions étant caractérisée par un coût et une capacité.

La sélection optimale est celle qui permet de maximiser le taux de production de la ligne sous des contraintes de budget, poids, espace, etc., ou de minimiser le coût total d'acquisition et d'exploitation sous des contraintes de taux de production, de disponibilité, etc. L'approche proposée se distingue des approches existantes par son pouvoir de *sélection à la fois des technologies des machines et des capacités des stocks.* Cette sélection conjointe des machines et des stocks conduit à un problème d'optimisation combinatoire encore plus difficile que celui d'allocation de stocks. En effet, le nouveau problème considéré ici résulte en fait du couplage de deux problèmes N-P difficiles, à savoir le problème d'allocation de la redondance et le problème d'allocation optimale des stocks [19].

Étant donné que certaines contraintes peuvent surgir comme la quantité limite de stocks entre machines à ne pas dépasser, le choix de technologies des machines et le niveau de redondance peuvent s'ajouter pour augmenter la performance des lignes de production. Les méthodes proposées dans la littérature requièrent un temps de calcul énorme lorsque les lignes de production sont de grande taille, ce qui rend leur intégration dans une approche globale permettant d'optimiser à la fois le choix de technologies de

machines et l'allocation des stocks tampons non praticable.

Les objectifs découlant de cette problématique peuvent ainsi être résumés comme suit :

- Proposer une méthode d'évaluation robuste et rapide de l'efficacité d'une ligne de production série-parallèle avec stocks tampons.
- Développer un algorithme de résolution pour traiter les problèmes de grande taille. Il s'agira d'une nouvelle méthode approchée capable de fournir, en un temps assez court, une solution très proche de l'optimum.

1.2.3 Conception optimale des systèmes séries-parallèles sans stocks tampons

Ce type de problème est plus connu dans la littérature sous le nom de problème d'allocation de la redondance ou « PAR ». Nous nous intéressons au cas où plusieurs machines sont connectées en parallèle (i.e., systèmes séries-parallèles). Le PAR est d'une extrême importance pour la conception des systèmes ayant des exigences en matière de fiabilité, comme c'est le cas de nombreux systèmes industriels comme les systèmes électroniques, les systèmes de télécommunication et les systèmes de puissance. Pour ces systèmes, la fiabilité est en fait considérée comme une mesure importante dans la conception. D'une manière générale, la conception d'un système nécessite le choix de technologies de composants parmi une liste disponible dans le marché. Ces technologies de composants sont caractérisées par leur fiabilité, coût, performance, poids, etc. Une stratégie est alors requise pour identifier la combinaison optimale des technologies à implanter dans le système ainsi que le niveau de redondance dans chaque sous-système. Cependant, ce type de problème est difficile à résoudre compte tenu du grand nombre de solutions possible. Misra [62], Agarwal et Barlow [2] ont montré que la plupart des problèmes d'optimisation de la fiabilité sont NP-difficiles. En 1992, Chern [19] a travaillé quant à lui sur la complexité des problèmes d'allocation de redondance dans les systèmes séries-parallèles, et a montré que pour la maximisation de la fiabilité du système sous une ou deux contraintes linéaires de ressource (coût et/ou poids), le problème d'allocation est NP-difficile.

Notre objectif par rapport à cette problématique d'allocation optimale de la redondance est de développer et tester une nouvelle méthode approchée de résolution pour résoudre des instances de grandes tailles. L'objectif étant de maximiser la fiabilité sous des contraintes de coût et de poids. Cette nouvelle méthode devra être caractérisée

par un temps de résolution relativement court et une très bonne qualité des solutions obtenues.

La figure 1.3 schématise les structures, les problèmes et les objectifs décrits dans cette thèse.

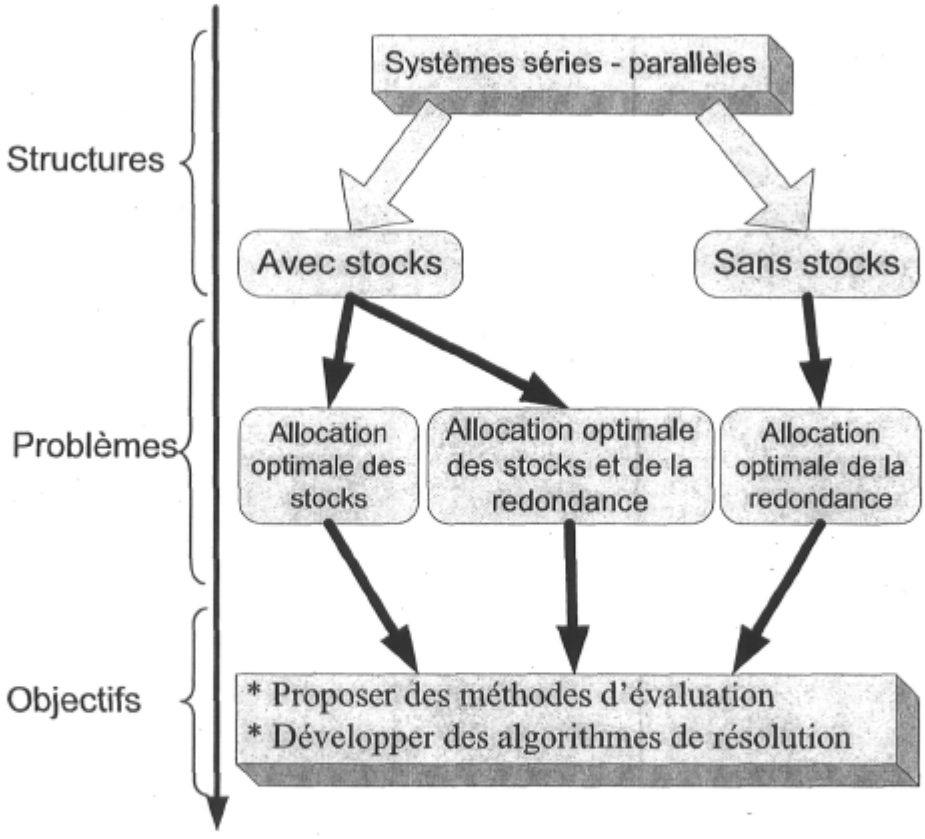

FIG. 1.3 – Structures, problèmes et objectifs

1.3 Revue des principaux travaux publiés

Nous allons classer notre revue de la littérature en trois catégories :

- Travaux publiés sur la conception optimale des systèmes séries-parallèles avec stocks tampons.

- Travaux publiés sur la conception optimale des systèmes séries-parallèles sans stocks tampons.
- Travaux sur les principales méthodes d'optimisation utilisables pour les problèmes combinatoires étudiés dans cette thèse.

1.3.1 Sur la conception optimale des systèmes séries-parallèles avec stocks tampons

Cette revue de la littérature sur les lignes séries-parallèles utilisant des stocks tampons sera divisée en deux parties. La première va concerner les méthodes d'évaluation de performance des lignes de production, alors que la deuxième partie sera dédiée aux algorithmes d'optimisation de ces lignes.

Méthodes d'évaluation de performance des lignes de production

Une revue de littérature détaillée sur les méthodes d'évaluation de performance des lignes de production peut être consultée par exemple dans [27] ou [67]. Plusieurs livres ont traité aussi ce problème [3][5][15][36][68][70]. C'est en 1953 que l'analyse des stocks tampons a fait ses premiers pas en 1953 suite aux travaux de Vladzievskii [89] qui a exploré le comportement d'une ligne de transfert automatique en utilisant la théorie des probabilités. En 1959, Koeningsberg [52] a élaboré la première technique analytique pour la modélisation des lignes de production avec stocks tampons. Dans son travail, il a mis l'accent sur les cas limites : stock tampon à capacité nulle, ou bien de capacité infinie.

L'approche des files d'attente pour l'analyse des lignes de production a été utilisée par plusieurs auteurs [13][52][53][70], etc. Des techniques de modélisation de systèmes manufacturiers sont proposées dans [14] où plusieurs types de files d'attente sont employés.

Buzacott a proposé en 1967, un modèle basé sur une approche Markovienne [16]. Dans son modèle il considère un stock de capacité finie, les distributions des pannes et des réparations sont supposées géométriques, et les pannes dépendent uniquement de l'opération et non du temps. Une extension de ce modèle a été faite par Buzacott et Shanthikumar [15], pour inclure aussi les pannes dépendant du temps. D'autres modèles pour traiter le cas de deux machines et un stock intermédiaire, ont été élaborés par plusieurs auteurs : Sheskin [74][75], Soyster [76] et autres. Gershwin et Berman

[37] ont étendu le modèle de Buzacott pour deux machines en prenant en compte les probabilités de défaillance et de réparation simultanées des deux machines.

Gershwin et Shick [38] ont proposé une méthode utilisant l'approche de chaînes de Markov pour analyser une ligne de production constituée de M machines non fiables et des stocks tampons à capacité finie. Ils ont appliqué leur méthode à une ligne comportant 3 machines. Dans leur méthode, ils ont supposé que toutes les machines ont des temps de traitement égaux et constants et qu'elles débutent leurs opérations en même temps.

Pour une longue ligne, les résultats exacts à base des chaînes de Markov sont très complexes à développer. La méthode d'agrégation permet l'étude des lignes de grandes tailles [83]. Dans cette méthode, chaque dipôle formé de deux machines et un stock tampon est agrégé par une seule machine équivalente. Une telle agrégation successive de dipôles conduit ainsi au remplacement d'une longue ligne par une seule machine équivalente possédant le même taux de production

D'autres approches de modélisation ont été explorées pour l'évaluation des paramètres de performance des systèmes de production utilisant des stocks tampons. Parmi ces approches qui ne s'inspirent ni de l'analyse des chaînes de Markov, ni de la théorie des files d'attente ou encore de la simulation, citons les travaux de Johri [49] qui a élaboré une approche utilisant la programmation linéaire pour estimer la capacité d'une ligne de production traitant plusieurs types de produits.

Gershwin [35] a proposé une autre méthode permettant de généraliser de façon approximative le résultat obtenu pour deux machines [37] à un système de n machines et $n-1$ stocks tampons. Cette technique se base sur la décomposition de la ligne originale, en plusieurs lignes fictives. L'élément clé de cette méthode de décomposition est d'assurer que les flots entrant et sortant des tampons des lignes fictives correspondent à ceux des tampons de la ligne de production originale.

Dallery, David et Xie [25] ont développé une méthode numérique itérative (appelée algorithme DDX) permettant d'approximer le taux de production d'une ligne à n machines et $(n-1)$ stocks tampons en se basant sur la méthode décomposition de Gershwin [35].

L'algorithme DDX [25] présente l'avantage de permettre d'approximer le taux de production de la ligne en un temps très court et les résultats générés s'approchent largement de ceux produits par simulation. Pour les lignes non équilibrées, les techniques d'homogénéisation proposées dans [26] ou l'algorithme ADDX de Burman [10]

représentent des extensions qui conservent les avantages de l'algorithme DDX.

Bien que la majorité des travaux existants s'intéressent à l'évaluation de performance d'une ligne de production en régime permanent, il existe aussi un certain nombre de travaux mettant l'emphase sur le comportement de la ligne en régime transitoire [4][20][31][38][40][87][88]. Plusieurs études basées sur la simulation ont également été réalisées [4][6][21][42][51], etc. Les modèles de simulation permettent généralement une prise en compte de contraintes plus réalistes que les modèles analytiques, et fournissent une analyse plus exhaustive que d'autres méthodes. Toutefois, beaucoup d'efforts peuvent être requis pour construire ces modèles et le temps de calcul est relativement élevé. Le temps requis pour le calcul du taux de production empêche l'utilisation de la simulation pour évaluer la fonction objective dans les problèmes d'optimisation combinatoire difficiles étudiés dans cette thèse. La prochaine partie présente une revue de littérature sur les méthodes d'optimisation des lignes de production.

Algorithmes d'optimisation des lignes de production

Il y a également une littérature assez importante sur l'optimisation des stocks intermédiaires. Une revue de littérature sur l'optimisation des stocks peut être trouvée par exemple dans [39].

En particulier, le problème d'optimisation de l'allocation des stocks intermédiaires a été traité par un grand nombre d'auteurs. Une étude détaillée de ces travaux de recherche a été fournie par [77] et [68]. La plupart de ces travaux concernent des lignes de production de petites tailles (voir par exemple [44] et [45]). Ho et al. [46] ont étudié l'optimisation de la performance d'une ligne de production en utilisant les techniques d'analyse des perturbations pour calculer des gradients par simulation. Gershwin et Shor [39] ont proposé des algorithmes basés sur la méthode du gradient. Dans [28], les auteurs ont developpé un algorithme génétique pour resoudre ce type de probleme. L'outil d'évaluation de performance utilisé est basé sur une méthode d'agrégation. Balakrishnan et al. [7] ont developpé un algorithme hybride à base de l'algorithme génétique pour la conception des lignes de production

Notons qu'il existe peu de travaux sur l'utilisation des techniques basées sur les métaheuristiques pour l'optimisation des lignes de production de grandes tailles [78][79][80]. Spinellis et al. [80] ont appliqué le recuit simulé pour resoudre le problème d'allocation optimale des stocks tampons. les auteurs ont utilisé une méthode basée sur les files d'attente pour l'évaluation de la performance. Dans [78], les auteurs ont developpé un

algorithme d'optimisation basé aussi sur le recuit simulé mais en utilisant la méthode de decomposition pour l'évaluation de la performance. Spinellis et Papadopoulos [79] ont presenté une étude comparative entre l'algorithme génétique et le recuit simulé pour la résolution du problème d'allocation optimale des stocks tampons dans des lignes dont les machines sont considérées fiables. Shi et Men [73] ont proposé une nouvelle méthode d'optimisation appelée *Nested Partitions* pour ce probleme. Les auteurs ont effectué des tests comparatifs entre cette méthode et la recherche avec tabou et les resultats obtenus avec la méthode *Nested Partitions* etaient supérieurs.

Notons aussi que les travaux existants considèrent souvent des lignes de production dont les machines sont fiables. De plus, ces études traitent le dimensionnement des stocks tampons, en supposant que les machines sont déjà sélectionnées. Ainsi, les seuls paramètres à trouver sont la taille des stocks tampons.

Nous rappelons, à ce propos que le problème d'allocation optimale de la redondance et des stocks que nous proposons de résoudre dans cette thèse se distingue des problèmes existants par le fait qu'il prenne en compte la sélection à la fois des technologies des machines et des capacités des stocks tampons et les lignes de production étudiées sont avec des machines non fiables et peuvent être homogènes ou non homogènes (i.e. les temps de traitement des machines sont différents). À notre connaissance il n'existe pas de travaux publiés traitant ce type de problème. Ce nouveau problème sera résolu en utilisant de nouvelles métaheuristiques telles que l'algorithme à colonies de fourmis et l'algorithme du grand déluge étendu.

1.3.2 Sur la conception optimale des systèmes séries-parallèles sans stocks tampons

Plusieurs techniques d'optimisation de la fiabilité s'intéressent à la détermination du nombre optimal d'unités redondantes dans chaque sous-système sont disponibles dans la littérature [81][84][85]. D'autres travaux s'intéressent particulièrement aux systèmes séries, séries-parallèles, stand-by et k parmi n. En particulier, une technique qui a prouvée son efficacité dans ce type de problème d'optimisation combinatoire est l'algorithme génétique [17][22][50][69][72][86][90]. Dans cette thèse, on s'intéressera particulièrement à la conception de nouvelles techniques à base de l'algorithme à colonies de fourmis [29][30] et l'algorithme du grand déluge étendu [9] pour la conception optimale des systèmes séries-parallèles sous des contraintes de budget et de poids.

Le problème de l'allocation optimale de la redondance est un problème NP-difficile

[19]. Plusieurs approches ont été utilisées pour résoudre ce problème [54][84]. Compte tenu de l'effort de calcul qui augmente d'une façon exponentielle avec la taille du problème et le nombre de contraintes, les approches traditionnelles restreignent généralement l'espace de solutions en affectant à chaque sous-système un seul type de technologie. Parmi les approches utilisant cette restriction, nous pouvons citer les techniques de programmation mathématique comme la programmation dynamique [32][64], la programmation en nombres entiers [8] ou mixte [47].

Plusieurs approches basées sur des métaheuristiques ont été développées pour résoudre ce problème où chaque sous-système peut contenir des machines avec différentes technologies. Coit et Smith [22] ont utilisé l'algorithme génétique Cependant, cet algorithme utilise une stratégie d'amélioration exigeant l'évaluation d'un grand nombre de solutions. Dans [56], les auteurs ont proposé une heuristique basée sur la recherche avec tabou où certains résultats obtenus par l'algorithme génétique ont été améliorés. Récemment, Liang et Smith [59] ont développé un algorithme basé sur l'algorithme à colonies de fourmis pour résoudre le problème d'allocation optimale de la redondance. Pour en valider l'efficacité, Liang et Smith [59] ont comparé leur algorithme avec l'algorithme génétique.

Dans cette thèse, nous proposons de concevoir un nouvel algorithme pour résoudre le problème d'allocation de la redondance. L'accent sera mis sur une comparaison détaillée avec les approches existantes dans la littérature.

1.3.3 Sur les méthodes d'optimisation

La plupart des problèmes d'optimisation combinatoire appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données. Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle et en intelligence artificielle. Ces méthodes peuvent être classées en deux grandes catégories : les méthodes exactes qui garantissent l'optimalité de la solution et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

Parmi les méthodes exactes, on trouve :

- la programmation dynamique consistant à placer le problème dans une famille de problèmes de même nature mais de difficulté différente puis à trouver une relation de récurrence liant les solutions optimales de ces problèmes.
- le Branch & Bound consistant à faire une énumération implicite en séparant le

problème en sous-problèmes et en évaluant ceux-ci à l'aide d'une relaxation (continue ou lagrangienne principalement) jusqu'à ne plus avoir que des problèmes faciles à résoudre ou dont on sait avec certitude qu'ils ne peuvent pas contenir de solution optimale.

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Malgré les progrès réalisés en matière de la programmation linéaire en nombres entiers, le temps de calcul nécessaire pour trouver une solution optimale risque d'augmenter exponentiellement avec la taille du problème, ce qui rend les méthodes exactes inefficaces face aux applications de taille importante.

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale. Parmi ces méthodes, il faut distinguer les heuristiques ciblées sur un problème particulier et les métaheuristiques plus puissantes et adaptables pour résoudre un grand nombre de problèmes. Elle est constituée d'un ensemble de concepts fondamentaux qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation donné. Ainsi les métaheuristiques sont adaptables et applicables à une large classe de problèmes. Cependant une métaheuristique, pour être suffisamment performante sur un problème donné nécessitera une adaptation plus ou moins fine. Grâce à ces métaheuristiques, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classiques de plus grande taille et pour de très nombreuses applications qu'il était impossible de traiter auparavant[58][65]. On constate, depuis ces dernières années, que l'intérêt porté aux métaheuristiques augmente continuellement en recherche opérationnelle et en intelligence artificielle. L'engouement pour les métaheuristiques est attribuable à plusieurs explications. Une des premières motivations pour ce champ de recherche provient de la pratique [63]. En effet, les contextes réels présentent des problèmes de gestion complexes devenant difficiles à traiter et les ressources disponibles sont souvent limitées. Le développement de ce domaine est également attribuable à la théorie de la complexité qui fournit une base rationnelle à l'utilisation de ces méthodes [33]. Enfin, les nouvelles possibilités informatiques expliquent aussi ce phénomène.

Les méthodes approchées peuvent se classer en différentes catégories :

- Constructives (algorithmes glouton, méthode Pilote, GRASP).
- Recherche locale (algorithmes de descente, recuit simulé, recherche Tabou,..).
- Évolutionnistes (algorithmes génétiques, algorithmes d'évolution, algorithme à colonies de fourmis).
- Réseaux de neurones (Modèle de Hopfield, machine de Boltzmann, réseau auto-

adaptatif).
- Heuristiques Bayésiennes (optimisation globale, optimisation discrète).
- Superposition (perturbation des données, perturbation des paramètres d'une heuristique).

Dans cette thèse, nous nous intéressons particulièrement à trois méthodes d'optimisation :

- L'algorithme à colonies de fourmis [29][30]. Cet algorithme a prouvé son efficacité et sa supériorité dans de nombreux problèmes d'optimisation combinatoire. Il sera appliqué surtout dans les problèmes de la redondance.
- L'algorithme étendu du grand déluge [9]. Cette méthode se caractérise par sa rapidité de convergence et ne requiert qu'un seul paramètre à régler. Elle sera appliquée au problème d'allocation des stocks tampons dans le cas où on ne dispose pas de versions de stocks à sélectionner mais une quantité finie à allouer à chaque zone tampon.
- L'algorithme du recuit simulé [55]. Cet algorithme est appliqué au problème d'allocation des stocks tampons.

Algorithme à colonies de fourmis Cet algorithme est basé essentiellement sur l'observation des fourmis qui construisent des chemins entre une source de nourriture et leur nid. Les fourmis sont capables de déposer sur leur chemin une certaine quantité de substance chimique volatiles, appelé phéromone, qu'elles peuvent détecter ensuite. Les fourmis se déplacent au hasard mais sont attirées par les chemins de phéromones déposées par d'autres fourmis. Ainsi plus les fourmis empruntent un chemin, plus il y aura de fourmis attirées par cet itinéraire.

Dans [29], la recherche du plus court chemin par une colonie de fourmis est illustrée par l'apparition d'un obstacle sur un chemin entre la source de nourriture et le nid. La présence d'un obstacle sur le chemin contraint les fourmis à en faire le tour par l'un des deux chemins. Au début, en moyenne, la moitié des fourmis choisissent le chemin le plus long et l'autre moitié le plus court. Etant donné que les fourmis déposent toutes des phéromones, le chemin le plus court sera plus marqué que le chemin le plus long pour un temps donné. Comme les fourmis suivent en probabilité le chemin le plus marqué par les phéromones, le phénomène s'amplifie et le chemin le plus court devient majoritairement suivi par les fourmis. Ainsi, l'aspect probabiliste du déplacement des fourmis assure qu'elles seront toujours à la recherche d'une meilleure solution car même quand les fourmis choisissent majoritairement le chemin le plus court, la probabilité de choisir l'autre chemin ne devient pas nulle. De plus, les phéromones étant des substances chi-

miques volatiles, elles s'évaporent avec le temps, ce qui permet aux fourmis de continuer l'exploration de leur environnement.

Cette nouvelle métaheuristique présente plusieurs caractéristiques :

- Versatile : elle peut être appliquée à des versions similaires du même problème comme dans le cas du problème asymétrique du voyageur de commerce qui est une extension du TSP.
- Robuste : elle peut être appliquée aux autres problèmes d'optimisation combinatoires comme le problème d'affectation quadratique (QAP) et les problèmes d'ordonnancement.
- C'est une approche basée sur la population.

L'algorithme à colonies de fourmis a été appliqué à plusieurs problèmes d'optimisation combinatoire et a prouvé son efficacité et la qualité des résultats obtenus. Parmi les applications, on peut citer : le problème du voyageur de commerce [29][30][82], le problème d'affectation quadratique [34][61], le problème d'ordonnancement des ateliers [23], le problème de tournée de véhicule [11][12], le problème de coloration de graphe [24].

À notre connaissance, l'algorithme à colonies de fourmis n'a jamais été appliqué aux problèmes d'optimisation de performances des lignes de production sauf dans le cas du problème d'allocation de la redondance [59] où les résultats numériques obtenus ont dépassés ceux de l'algorithme génétique [22].

Algorithme du grand deluge étendu L'algorithme du grand deluge étendu a été introduit par Burke et al.[9]. Il est considéré comme un algorithme de recherche locale où certaines « mauvaises » solutions dont la valeur ne dépasse pas une certaine limite L sont acceptées. Cette limite diminue (dans le cas de problèmes de minimisation) d'une façon monotone durant la recherche. La valeur initiale de L est égale à la fonction objective initiale. Le seul paramètre est alors à régler est celui du taux de dégradation L (noté ΔL).

L'avantage de cet algorithme est qu'on a besoin de régler qu'un seul paramètre ΔL. Les tests effectués dans [9] ont montré que le temps de convergence de l'algorithme dépend de la valeur de ΔL. En effet, en augmentant sa valeur (pour les problèmes de minimisation), le temps de convergence diminue mais la qualité des solutions peut se dégrader aussi.

La seule application réalisée de cette approche est celle du problème d'optimisation de l'horaire des examens (Exam Timetabling Problem). Les résultats ont prouvé l'efficacité de l'algorithme[9]. Plusieurs résultats existants ont été améliorés où des approches comme la recherche tabou avaient été utilisées.

Algorithme du recuit simulé L'algorithme du recuit simulé a été proposé, pour la première fois, par Kirkpatrick et al. [55] en s'inspirant de l'algorithme de (Metropolis et al.[60] , qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction f à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système.

Partant d'une solution donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'energie du système, soit celle-ci le dégrade. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

Comme pour toute métaheuristique, la méthode trouve son application dans de nombreux domaines dans lesquels on a à résoudre des problèmes d'optimisation difficiles. On peut citer entre autres les problèmes d'ordonnancement (e.g. [66][71]), les problèmes de routage (e.g. [18][41]) ou les problèemes d'amenagement d'usines (e.g. [48][57]).

1.4 Résumé de la méthodologie et des contributions

Afin de répondre aux objectifs de cette thèse, nous avons poursuivi une méthodologie basée sur les deux étapes générales suivantes :

Étape 1 : Méthode d'évaluation
- Analyser les différentes méthodes existantes dans la littérature et sélectionner pour chacun des problèmes la méthode appropriée.
- Développer, le cas échéant, une nouvelle approche d'évaluation de performance des systèmes étudiés en utilisant les principes de décomposition, d'agrégation des machines parallèles et d'homogénéisation.
- Procéder à la validation par la simulation de toute nouvelle méthode d'évaluation.

Étape 2 : Méthode d'optimisation

- Analyser les différentes méthodes d'optimisation existantes dans la littérature et sélectionner pour chacun des problèmes la méthode appropriée.
- Tester les nouveaux algorithmes développés sur des instances de problèmes issues de la littérature ou générées aléatoirement.

Nous allons maintenant expliquer comment nous avons appliqué chacune de ces deux étapes aux trois problèmes étudiés.

Problème 1 : Allocation optimale des stocks

Étape 1 : Méthode d'évaluation
- Analyser les méthodes d'évaluation des performances existantes dans la littérature des lignes de production homogènes et non homogènes dont les machines sont assujetties à des défaillances aléatoires.
- Appliquer la méthode de décomposition et l'algorithme DDX pour l'évaluation des lignes de production.

Étape 2 : Méthode d'optimisation
- Analyser les différentes méthodes d'optimisation existantes dans la littérature qui ont été appliquées sur le problème d'allocation optimale des stocks tampons, et notamment celles à base de métaheuristiques.
- Développer un algorithme efficace et robuste basé sur l'algorithme du grand déluge étendu [9].
- Effectuer une comparaison avec les approches existantes dans la littérature en générant aléatoirement des instances. Il s'agit notamment du recuit simulé [78] et l'algorithme génétique [79].

Problème 2 : Allocation optimale des stocks et de la redondance

Étape 1 : Méthode d'évaluation
- Analyser les méthodes d'évaluation des performances existantes dans la littérature des lignes de production de type séries-parallèles dont les machines sont assujetties à des défaillances aléatoires.
- Développer une nouvelle approche d'évaluation de performance des lignes séries-parallèles en utilisant les principes de décomposition, d'agrégation des machines parallèles et d'homogénéisation.
- Procéder à la validation par la simulation de cette nouvelle méthode d'évaluation sur un grand nombre d'instances générées aléatoirement. Les tests seront effectués sur des lignes de taille différente.

Étape 2 : Méthode d'optimisation
- Analyser les différentes méthodes d'optimisation existantes dans la littérature.
- Développer un algorithme basé sur l'algorithme à colonies de fourmis [29] couplé avec une procédure d'amélioration pour la conception optimale des lignes de production séries-parallèles.
- Développer un algorithme basé sur l'algorithme de recuit simulé [55] pour la conception optimale des lignes de production séries-parallèles.
- Effectuer une comparaison entre les deux algorithmes en générant aléatoirement des instances de grande taille. Ces instances sont générées parce que le problème formulé est nouveau et il n'existe donc pas d'instances dans la littérature.

Problème 3 : Allocation optimale de la redondance

Étape 1 : Méthode d'évaluation
- L'évaluation de la fiabilité des systèmes séries-parallèles sans stocks tampons peut être facilement calculée.

Étape 2 : Méthode d'optimisation
- Analyser les différentes méthodes d'optimisation existantes dans la littérature qui ont été appliquées sur le problème d'allocation optimale de la redondance.
- Développer un algorithme hybride efficace et robuste basé sur une hybridation de l'algorithme à colonies de fourmis [29] avec l'algorithme du grand déluge étendu [9].
- Effectuer une comparaison de notre nouvelle approche hybride avec les différentes approches existantes dans la littérature, et ce sur des instances de problèmes déjà existantes.

1.5 Conclusion

Le présent chapitre a commencé par introduire la problématique générale et les objectifs de la thèse. Ensuite, une revue des principaux travaux publiés et un résumé de la méthodologie ont été présentés. Les trois prochains chapitres présentent trois contributions originales au domaine de la conception optimale des systèmes de production avec et sans stockes tampons.

Bibliographie

[1] Abdul-Kader W. and Ait-Kadi D. *Buffer zone allocation : an alternative approach.* 12th International Conference on CAD/CAM Robotics and Factories of the Future, London,England, August 14-16, 1996.

[2] Agarwal A. and Barlow R.E. *A Survey of Network Reliability and Domination Theory.* Operations Research, 1984, 32, 478-492.

[3] Altiok T. *Performance Analysis of Manufacturing Systems.* Springer, New-York, 1997.

[4] Anderson D.R. and Moodie C.L *AOptimal buffer storage capacity in productin line systems.* International Journal of Production Research, 1969, 7, 233-240.

[5] Askin R. and Standridge C. *Modeling and Analysis of Manufacturing Systems.* Wiley, New York, NY, 1993.

[6] Baker K.R., Powell S.G. and Pyke D.F. *Buffered and unbuffered assembly systems with variable processing times.* Journal of Manufacturing and Operations Management, 1990, 3, 200-223.

[7] Balakrishnan, P. V., R. Gupta, and Jacob V.*Development of Hybrid Genetic Algorithms for Product Line Designs.* IEEE Transactions on Systems, Man, and Cybernetics, 2004, 34(1), 468-483.

[8] Bulfin R.L and Liu C.Y.*Optimal Allocation of Redundant components for Large Systems.* IEEE Transactions on Reliability, 1985, R-34, 241-247.

[9] Burke EK, Bykov Y, Newall JP and Petrovic S. *A time-predefined local search approach to exam timetabling problems.*IIE Transactions, 2004, 36(6), 509-28.

[10] Burman M.H. *New Results in Flow Line Analysis.* Ph. D. Thesis, MIT, Cambridge MA., 1995.

[11] Bullnheimer B., Hartl R.F. and Strauss C.*Applying the Ant System to the Vehicle Routing Problem.* In Meta-heuristics : Advances and Trends in Local Search for Optimization, Voss S., Martello S., Osman I.H. and Roucairol C. (eds.), Kluwer Academic Publishers, Boston, 1999, 285-296.

[12] Bullnheimer B., Hartl R.F. and Strauss C. *An improved ant system algorithm for the ve-hicle routing problem.* Annals of Operations Research, 1999, 89, 319-328.

[13] Buxey G.M., Slack N.D. and Wild R. *Production flow line system design -a review.* AIIE Transactions, 1973, 5, 37-48.

[14] Buzacott, J.A and Shanthikumar J.G. *Design of manufacturing systems using queueing models.* Queueing System, 1992, 12, 135-213.

[15] Buzacott J.A and Shanthikumar J.G. *Stochastic Models of Manufacturing Systems.* Prentice Hall, Englewood Cliffs, N.J, 1993.

[16] Buzacott J.A. *Automatic transfer lines with buffer stocks.* International Journal of Production Research, 1967, 5(3), 182-200.

[17] Cantoni M., Marseguerra M. and Zio E. *Genetic algorithms and Monte Carlo simulation for optimal plant design.* Reliability Engineering and System Safety, 2000, 68, 29-38.

[18] Cerny V.*Thermodynamical Approach to the Traveling Salesman Problem : An Efficient Simulation Algorithm.* Journal of Optimization Theory and Applications, 1985, 45(1), 41-51.

[19] Chern MS. *On the computational complexity of reliability redundancy allocation in a series system.* Operations Research Letter 1992 ;11 :309-15.

[20] Ciprut P, Hongler M-O and Salama Y. *On the variance of the production output of transfer lines.* IEEE Transactions on Robotics and Automation, 1999, 15(1), 33-43.

[21] Conway R., Maxwell W., McClain J.O. and Thomas L.J.*The role of work in process inventory in serial production lines.* Operations Research, 1988, 36(2), 229-241.

[22] Coit DW and Smith AE. *Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm.* IEEE Transactions on Reliability, 1996, 45(2), 254-60.

[23] Colorni A., Dorigo M., Maniezzo V., and Trubian M.*Ant system for job-shop scheduling.* JORBEL - Belgian Journal of Operations Research,Statistics and Computer Science, 1994, 34, 39-53.

[24] Costa D. and Hertz A.*Ants can colour graphs.* Journal of the Operational Research Society, 1997, 48, 295-305.

[25] Dallery Y, David R and Xie X.L. *An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers.* IIE transactions, 1988, 20(3), 280-283.

[26] Dallery Y and Le Bihan H. *Homogenisation techniques for the analysis of production lines with unreliable machines having different speeds.* European Journal of Control, 1997, 3, 200-215.

[27] Dallery Y and Gershwin S.B. *Manufacturing flow line systems : a review of models and analytical results.* Queuing Systems theory and Applications, Special Issue on Queuing Models of Manufacturing Systems, 1992, 12(1-2), 3-94.

[28] Dolgui, A., Ereemev, A., Kolokolov, A. and Sigaev, V.*A genetic algorithm for allocation of buffer storage capacities in production line with unreliable machines.* Journal of Mathematical Modelling and Algorithms, 2002, 1, 89-104.

[29] Dorigo M, Maniezzo V and Colorni A. *The Ant System : Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man and Cybernetics- Part B, 1996, 26(1), 1-13.

[30] Dorigo M and Gambardella LM. *Ant colony system : a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation, 1997, 1(1), 53-66.

[31] Elsayed E.A. and Lin B.W.*Transient Behavior of Ordered Entry Multi-Channel Queueing Systems.* International Journal of Production Research, 1980, 18(4), 491-501.

[32] Fyffe D.E, Hines W.W and Lee N.K. *System Reliability Allocation And a Computational Algorithm.* IEEE Transactions on Reliability, 1968, R-17(2), 64-69.

[33] Garey M.R. and Johnson D.S.*Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H Freeman and Company, 1979.

[34] Gambardella L.M, Taillard E. and Dorigo M.*Ant colonies for the Quadratic Assignment Problem.* Journal of the Operational Research Society, 1999, 50,167-176.

[35] Gershwin S.B. *An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking.* Operations Research, 1987, 35, 291-305.

[36] Gershwin S.B. *Manufacturing systems engineering.* Prentice-Hall, 1994.

[37] Gershwin S.B. and Berman O. *Analysis of transfer lines consisting of two unreliable machines with random processing times and finite sorage buffers.* AIIE Transactions, 1981, 13, 2-11.

[38] Gershwin S.B. and Schick I.C. *Modeling and analysis of three-stage transfer lines with unreliable machine and finite buffers.* Operations Research, 1983, 31, 354-380.

[39] Gershwin S.B and Schor J.E. *Efficient algorithms for buffer space allocation.* Annals of Operations Research, 2000, 93, 117-144.

[40] Gupta U.C. and Sharma O.P. *On the transcient behavior of a model for queues in series with finite capacity.* International Journal of Production Research, 1983, 21, 6, 869-879.

[41] Golden B.L. and Skiscim C.C *Using Simulated Annealing to Solve Routing and Location Problems.* Naval Research Logistics Quarterly, 1986, 33, 261-279.

[42] Hanifin E.*Increased transfer line productivity utilizing systems simulation.* Ph.D Thesis, 1975, University of Detroit.

[43] Huang M.-G, Chang P.-L and Chou Y.-C.*Buffer allocation in flow-shop-type production systems with general arrival and service patterns.*Computers and operations research,2002, 29(2), 103-121.

[44] Hillier F.S, So K.C. *The effect of the coefficient of variation of operation times on the allocation of storage space in production line system.* IIE Transactions, 1991, 23, 198-206.

[45] Hillier F.S, So K.C, Boling R.W. *Notes : Toward characterizing the optimal allocation of storage space in production line systems with variable processing times.* Management Science, 1993, 39, 126-133.

[46] Ho Y.C, Eyler M.A, Chien T.T. *A Gradient Technique for General Buffer Storage Design in a Production Line.* International Journal of Production Research, 1979, 17, 557-580.

[47] Hwang C.L., Tillman F.A. and Kuo W *Reliability optimization by generalized Lagrangian-function and reduced-gradient methods.* IEEE Transactions on Reliability, 1979, R-28,316-319..

[48] Jajodia S., Minis I., Harhalakis G. and Proth J-M.*CLASS : Computerized LAyout Solutions Using Simulated Annealing.* International Journal of Production Research, 1992, 30(1), 95-108.

[49] Johri, P.K.*A linear programming approach to capacity estimation of automated production lines with finite buffers.* International Journal of Production Research, 1987, 25(6), 851-866.

[50] Joyce, P.A., Withers, T.A. and Hickling P.J. *Application of genetic algorithms to optimum offshore plant design.* Proceedings of ESREL 98, 1998, Trondheim (Norway), June 16-19, 665-671.

[51] Kay E. *Buffer stocks in automatic transfer lines.* International Journal of Production Research, 1972, 10(2), 155-165.

[52] Koenisberg E. *Production lines and internal storage - a review.* Management Science,1959, 5, 410-433.

[53] A.D. Knott A.D. *The inefficiency of a series of work stations - a simple formula.* International Journal of Production Research, 1970, 8, 109-119.

[54] Kuo W and Prasad VR. *An annotated overview of system-reliability optimization.* IEEE Transactions on Reliability, 2000, 49(2), 176-87.

[55] Kirkpatrick S.,Gelatt C.D. and Vecchi M.P.*Optimization by Simulated Annealing.* Science, 1983, 220, 671-680.

[56] Kulturel-Konak S, Smith AE and Coit D.W. *Efficiently solving the redundancy allocation problem using tabu search.* IIE transactions, 2003, 35, 515-26.

[57] Kouvelis P., Chiang W.-C., and Fitzsimmons J. *Simulated annealing for machine layout problems in the presence of zoning constraints.* European Journal of Operational Research, 1992, 57, 190-202.

[58] Laporte, G. and Osman I.*Metaheuristics in combinatorial optimization.* Annals in Operations Research,1996, 63.

[59] Liang Y.C and Smith A.E. *An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP).* IEEE transactions on Reliability, 2004, 53 (3), 417-23.

[60] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E.*Equation of State Calculations by Fast Computing Machines.* Journal of Chemical Physics, 1953, 21(6), 1087-1092.

[61] Maniezzo V. and Colorni A.*The Ant System Applied to the Quadratic Assignment Problem.* IEEE Transactions on Knowledge and Data Engineering, 1999, 11(5), 769-778.

[62] Misra KB and Sharma U. *An Efficient Algorithm to Solve Integer-Programming Problems Arising in System-Reliability Design.* IEEE Transactions on Reliability, 1991, 40(1), 81-91.

[63] MacCarthy B.L. and Liu J.*Addressing the Gap in Scheduling Research : A Review of Optimization and Heuristic Methods in Production Scheduling.* International Journal of Production Research, 1993, 31(1), 59-79.

[64] Nakagawa Y and Miyazaki S. *Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints.* IEEE Transactions on Reliability, 1981, R-30(2), 175-80.

[65] Osman I.H. and Kelly J.P.*Meta-heuristics : theory and applications.* Kluwer, Boston, 1996.

[66] Osman I.H. and Potts C.N.*Simulated annealing for permutation flow-shop scheduling.* Omega, 1989, 17, 551-557.

[67] Papadopoulos, H.T. and Heavey C. *Queueing theory in manufacturing systems analysis and design, A classification of models for production and transfer lines.* European Journal of Operational Research, 1996, 1, 1-27.

[68] Papadopoulos H.T, Heavey C, Browne J. *Queueing Theory in Manufacturing Systems Analysis and Design.* Chapman and Hall, London, 1993.

[69] Painton L and Campbell J. *Genetic Algorithms in Optimization of System Reliability.* IEEE Transactions on Reliability, 1995, 44(2), 172-78.

[70] Perros H. *Queueing Networks with Blocking.* Oxford, 1994.

[71] Potts C.N. and Van Wassenhove L.N.*Single machine tardiness sequencing heuristics.* IIE Transactions, 1991, 23, 346-354.

[72] Ramachandran V., Sivakumar V. and Sathiyanarayanan K.*Genetics based redundancy optimization.* Microelectronics and Reliability, 1997, 37(4), 661-663.

[73] Shi L and Men S. *Optimal buffer allocation in production lines.* IIE Transactions, 2003, 35, 1-10.

[74] Sheskin T.J. *Allocation of Interstage Storage along an Automatic Production Transfer Line with Discret Flow.* Unpublished Ph.D thesis, Departement of Industrial and Management Systems Engineering, Pennsylvenia State University, 1974.

[75] Sheskin T.J. *Allocation of Interstage Along an Automatic Production Line.* AIIE transactions, 1976, 8, 146-152.

[76] Soyster A.L., Schmidt J.W. and Rohrer M.W. *Allocation of Buffers Capacities for a Class of Fixed Cycle Production Lines.* AIIE transactions, 1979, 11,2.

[77] Singh A, MacGregor Smith J. *Buffer allocation for an integer nonlinear network design problem.* Computers & Operations Research, 1997, 24, 453-472.

[78] Spinellis D, Papadopoulos C.T. *A simulated annealing approach for buffer allocation in reliable production lines.* Annals of Operations Research, 2000, 93, 373-384.

[79] Spinellis D, Papadopoulos C.T. *Stochastic Algorithms for Buffer allocation in Reliable Production Lines.* Mathematical Problems in Engineering, 2000, 5, 441-458.

[80] Spinellis D, Papadopoulos C, Macgregor Smith J. *Large production line optimization using simulated annealing.* International Journal of Production Research, 2000, 38, 509- 541.

[81] Sung C.S. and Lee H.K. *A Branch-and-Bound Approach for Spare Unit Allocation in a Series System.* European Journal of Operational Research, 2004, 75, 217-232.

[82] Stutzle T. and Hoos H.H.*MAX-MIN Ant System.* Future Generation Computer Systems, 2000, 16(8), 889-914.

[83] Terracol C. and David R. *An aggregation method for performance valuation of transfer lines with unreliable machines and finite buffers.* IEEE International Conferance on Robotics and Automation, 1987, Raleigh, NC.

[84] Tillman FA, Hwang CL and Kuo W. *Optimization techniques for system reliability with redundancy- a review.* IEEE Transaction on Reliability, 1977, R-26(3), 147-155.

[85] Tillman FA, Hwang C-L and Kuo W. *Determining Component Reliability and Redundancy for Optimum System Reliability.* IEEE Transactions on Reliability, 1977, R-26(3), 162-165.

[86] Yang J., Hwang M., Sung T. and Jin Y.*Application of genetic algorithm for reliability allocation in nuclear power plant.* Reliability Engineering and System Safety, 1999, 65(3), 229-238.

[87] Wilhelm W.E. and Sastri T. *An investigation of Flow Line Operating Characteristics during Start-up*. International Journal of Production Research, 1979, 17(4), 345-358.

[88] Wilhelm W.E.*A model to approximate to approximate transcient performance of the flow shop*. International Journal of Production Research, 1986, 24(1), 33-50.

[89] Vladzievskii A.P. *Losses of working time and the division of automatic lines into sections*. Stanki i Instrument 24 (in Russian), 1953.

[90] Zhao R. and Liu B. *Stochastic programming models for general redundancy-optimization problems*. IEEE Transactions on Reliability, 2003, 52, 181-191.

Chapitre 2

Hybrid algorithm for the redundancy allocation problem

Résumé

Le problème d'allocation de la redondance (PAR) est un problème NP-difficile qui consiste à sélectionner, pour un système série-parallèle, la version et le nombre d'éléments à mettre en parallèle dans chaque sous-système en maximisant la fiabilité totale du système sous différentes contraintes. Ce chapitre présente un algorithme hybride très efficace pour résoudre ce problème d'optimisation. Cet algorithme est basé sur une hybridation de l'algorithme a colonies de fourmis avec l'algorithme du grand déluge étendu. Les résultats numériques obtenus des 33 problèmes tests ont montré que l'algorithme proposé est très compétitif par rapport aux autres approches présentées dans la littérature.

2.1 Abstract

The redundancy allocation problem (RAP) is a well known NP-hard problem which involves the selection of elements and redundancy levels to maximize system reliability given various system-level constraints. As telecommunications and internet protocol networks, manufacturing and power systems are becoming more and more complex, while requiring short developments schedules and very high reliability, it is becoming increasingly important to develop efficient solutions to the RAP. This paper presents an efficient algorithm to solve this reliability optimization problem. The idea of a heuristic approach design is inspired from the ant colony meta-heuristic optimization method and the degraded ceiling local search technique. Our hybridization of the ant colony meta-heuristic with the degraded ceiling performs well and is competitive with the best-known heuristics for redundancy allocation. Numerical results for the 33 test problems from previous research are reported and compared. The solutions found by our approach are all better than or are in par with the well-known best solutions.

2.2 Introduction

2.2.1 Problem description

The redundancy allocation problem (RAP) is a well known combinatorial optimization problem where the design goal is achieved by discrete choices made from elements available on the market. The system consists of n components in series (figure 2.1). For each component i ($i = 1, 2, \ldots, n$) there are various versions of elements, which are proposed by the suppliers on the market. Elements are characterized by their cost and weight according to their version. Each component i contains a number of elements connected in parallel. Different elements can be placed in parallel. A component i is functioning properly if at least k_i of its p_i elements are operational (k-out-of-n :G).

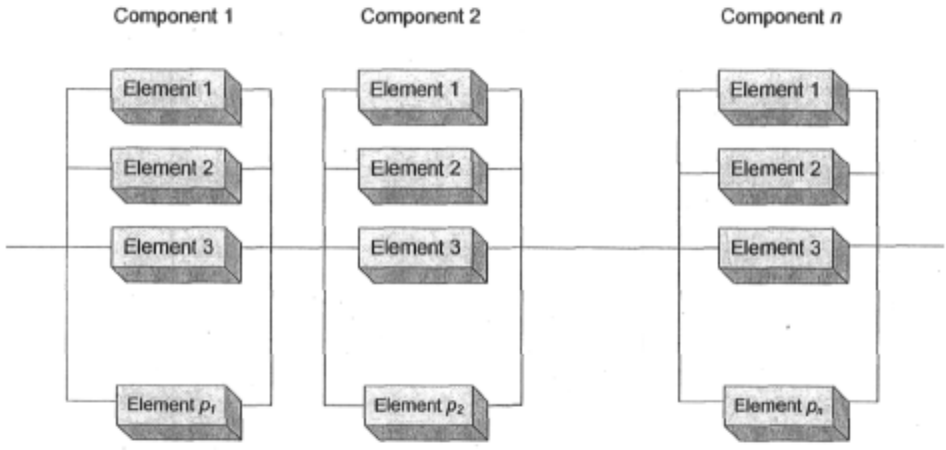

FIG. 2.1 – Series-parallel system structure

The series-parallel system sketched in figure 2.1 is a logic-diagram representation for many design problems encountered in industrial systems. As it is pointed out in [9] and [22], electronics industry is an example where the RAP is very important. In fact, in this industry most systems require very high reliability and the products are usually assembled and designed using off-the-shelf elements (capacitors, transistors, microcontrollers, etc.) with known characteristics. Other examples where the above type of structure is becoming increasingly important include telecommunications systems and power systems. In all these systems, redundancy is indeed a necessity to reach the required levels of reliability and the RAP studied in this paper is therefore one of the major problems inherent to optimal design of reliable systems.

Assumptions

1. Elements and the system may experience only two possible states : good and failed.
2. The system weight and cost are linear combinations of element weight and cost.
3. The element attributes (reliability, cost and weight) are known and deterministic.
4. Failed elements do not damage the system, and are not repaired.
5. All redundancy is active : failure rates of elements when not in use are the same as when in use.
6. The supply of elements is unlimited.
7. Failures of individual elements are s-independent.

Notation

R_{sys}	overall reliability of the series-parallel system
$R*$	optimal solution
C	cost constraint
W	weight constraint
n	number of components
i	index for components
a_i	number of available elements choices (i.e.,versions) for component i
r_{ij}	reliability of element j available for component i
w_{ij}	weight of element j available for component i
c_{ij}	cost of element j available for component i
x_{ij}	number of element j used in component i
$\mathbf{x}_i$	$(x_{i1}, x_{i2}, ..., x_{ia_i})$
p_i	total number of elements used in component i
$p_{\max}$	maximum number of elements in parallel
k_i	minimum number of elements in parallel required for component i
$\mathbf{k}$	$(k_1, k_2, ..., k_n)$
$R_i(\mathbf{x}_i \| k_i)$	reliability of component i, given k_i.
$C_i(\mathbf{x}_i)$	total cost of component i
$W_i(\mathbf{x}_i)$	total weight of component i

The RAP is formulated to maximize system reliability given restrictions on system cost and weight. That is,

$$\text{Maximize} \qquad R_{sys} = \prod_{i=1}^{n} R_i(\mathbf{x}_i|k_i) \tag{2.1}$$

$$\text{Subject to} \qquad \sum_{i=1}^{n} C_i(\mathbf{x}_i) \leq C \tag{2.2}$$

$$\sum_{i=1}^{n} W_i(\mathbf{x}_i) \leq W \tag{2.3}$$

Constraints (2.2) and (2.3) represent respectively the budget and the weight constraints. If there is a pre-selected maximum number of elements which are allowed in parallel, the following constraint (2.4) is added :

$$k_i \leq p_i \leq p_{\max} \qquad \forall\, i = 1, ..., n \tag{2.4}$$

2.2.2 Prior literature

The RAP is NP-hard [8] and has previously been solved using many different optimization approaches and for different formulations as summarized in [32], and more recently in [23]. Optimization approaches to determine optimal or very good solutions for the RAP include dynamic programming, e.g. [2][16][28][34], mixed-integer and nonlinear programming, e.g. [33], and integer programming, e.g. [4][18][19][26] Nevertheless, these methods are limited to series-parallel structures where the elements used in parallel are identical. This constitutes a drawback since in practice many systems designs use different elements performing the same function, to reach high reliability level [22]. For example, as explained in [9], (airplanes use a primary electronic gyroscope and a secondary mechanical gyroscope working in parallel, and most new automobiles have a redundant (spare) tire with different size and weight characteristics forming a 4-out-of-5 :G standby redundant system). Because of the above-mentioned drawback and of the exponential increase in search space with problem size, heuristics have become a popular alternative to exact methods. Meta-heuristics, in particular, offer flexibility and a practical way to solve large instances of the relaxed RAP where different elements can be placed in parallel.

Genetic algorithm (GA) is a well-known meta-heuristic used to solve combinatorial reliability optimization problems [9][24][30][35][36]. In addition to genetic algorithms, other heuristic or meta-heuristic approaches have also been efficiently used to deal with

system reliability problems. A tabu search (TS) meta-heuristic [21] has been developed in [22] to efficiently solving the RAP, while the ant colony optimization meta-heuristic [11] is used in [25] to solve the problem.

In light of the aforementioned approaches, this paper presents a heuristic approach to solve the RAP. This paper combines an ant colony optimization approach and a degraded ceiling local search technique. This approach is said to be hybrid and will be called ACO/DC (for Ant Colony Optimization and Degraded Ceiling).

The idea of employing a colony of cooperating agents to solve combinatorial optimization problems was recently proposed in [12]. The ant colony approach has been successfully applied to the classical traveling salesman problem [13][14], to the quadratic assignment problem [17], and to scheduling [1][3] Ant colony shows very good results in each applied area. The ant colony has also been adapted with success to other combinatorial optimization problems such as the vehicle routing problem [5], telecommunication networks management [15], graph coloring [10], constraint satisfaction [31] and Hamiltonian graphs [37]. In [27], the authors used ant system to solve the optimal design problem of series system under budget constraints. The ant colony approach has also been applied for the RAP of multi-state systems in [29]. For the RAP in the binary state case, which is the focus of the present paper, the only existing work is that of [25].

Unlike [25], we hybridize the ant colony meta-heuristic with a variant of a local search meta-heuristic, called the degraded ceiling approach. The degraded ceiling approach was recently introduced in [6], where it has been tested real-world university examination timetabling problems and the experiments have confirmed its high effectiveness. Furthermore, our solution encoding and solution construction method is different from that of [25].

2.2.3 Paper outline

The remainder of this paper is organized as follows. In section 2.3, we describe the proposed heuristic approach. We first present the proposed ACO algorithm. After this, we present the degraded ceiling algorithm and how it was adapted to the RAP within the ACO framework. In Section 2.4, the proposed approach is implemented on the 33 variations of [16]. The numerical results obtained from our hybrid ACO/DC approach (i.e., combining ant colony optimization and degraded ceiling) shows that all solutions are all better than or tie the best-published results from the literature [9][20][22][25][28]. Comparisons are summarized in the conclusion of section 2.6.

2.3 Hybrid solution approach : ACO/DC

2.3.1 The ACO algorithm

The principle

Ants lay down in some quantity an aromatic substance, known as pheromone, in their way to food. The pheromone quantity depends on the length of the path and the quality of the discovered food source. An ant chooses a specific path in correlation with the intensity of the pheromone. The pheromone trail evaporates over time if no more pheromone is laid down. Other ants can observe the pheromone trail and are attracted to follow it. Thus, the path will be marked again and will therefore attract more ants. The pheromone trail on paths leading to rich food sources close to the nest will be more frequented and will therefore grow faster. In that way, the best solution has more intensive pheromone and higher probability to be chosen. The described behaviour of real ant colonies can be used to solve combinatorial optimization problems by simulation : artificial ants searching the solution space simulate real ants searching their environment. The objective values correspond to the quality of the food sources. The ACO approach associates pheromone trails to features of the solutions of a combinatorial problem, which can be seen as a kind of adaptive memory of the previous solutions. In addition, the artificial ants are equipped with a local heuristic function to guide their search through the set of feasible solutions. Solutions are iteratively constructed in a randomized heuristic fashion biased by the pheromone trails left by the previous ants. The pheromone trails are updated after the construction of a solution, enforcing that the best features will have a more intensive pheromone.

Applying the ACO meta-heuristic to the RAP

The general algorithm

To apply the ACO meta-heuristic to a combinatorial optimization problem, it is convenient to represent the problem by a graph $G = (\varsigma,\Lambda)$, where ς are the nodes and Λ is the set of edges. To represent our problem as such a graph, we introduce the following sets of nodes and edges :

- Three sets of nodes :

1. The first set of nodes ($N1$) represents the components.
2. The second set of nodes ($N2$) represents, for each component, the numbers of elements which can be used in parallel. For example, if the maximum number of elements in parallel is three ($p_{max} = 3$), the set N2 will be given by three nodes corresponding to one element, two parallel elements and three parallel elements.
3. The third set ($N3$) represents the versions of elements available for each component.

- Two sets of edges :

1. The first set of edges is used to connect each component node in the set $N1$ to the corresponding nodes in $N2$.
2. The second set of edges is used to connect the nodes in $N2$ to the nodes in $N3$ of their available versions.

Figure 2.2 illustrates such a graph for a simple trivial case of one component, with $p_{max} = 3$ and two available versions.

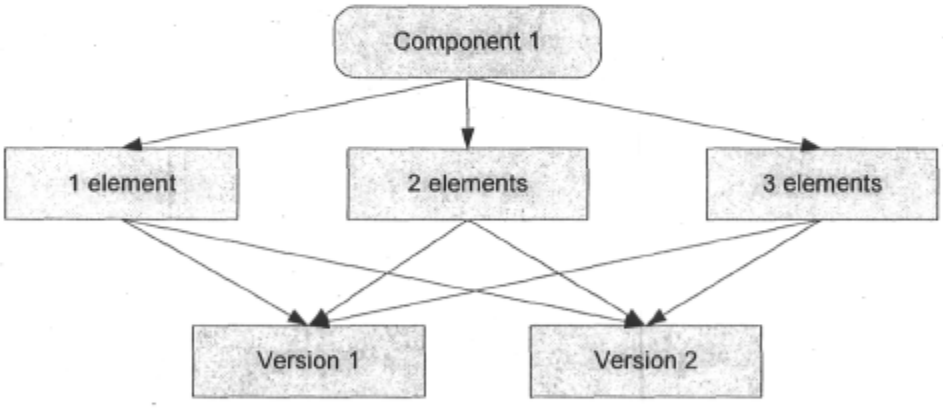

FIG. 2.2 – A graph example for one component with $p_{max} = 3$ and two available versions

Informally, our algorithm works as follows : m ants are initially positioned on a node representing a component. Each ant represents one possible structure of the entire system. This entire structure is defined by the vectors x_i ($i = 1, \ldots, n$). Each ant builds a feasible solution (called a tour) to our problem by repeatedly applying stochastic greedy rules (i.e., the state transition rules). Once all ants have terminated their tour, the amount of pheromone on edges is modified by applying the global updating rule. Ants are guided, in building their tours, by both heuristic information (they prefer to choose "less expansive" edges), and by pheromone information. Once an ant has built a structure, the obtained solution is improved by using a local search algorithm. This step is performed only in the following cases :

- the obtained solution by the ant is feasible and,
- the quality of the solution is "good". The term "good" means here that the reliability R_{sys} of the structure should be either better than the best solution $R*$, i.e.,$R_{sys} \geq R^*$, or close to this best solution R^*, i.e.,$\frac{R^*-R_{sys}}{R^*} \leq +l$ where l represents the solution quality level.

Figure 2.3 presents an overview of an iteration of the algorithm needed for each ant.

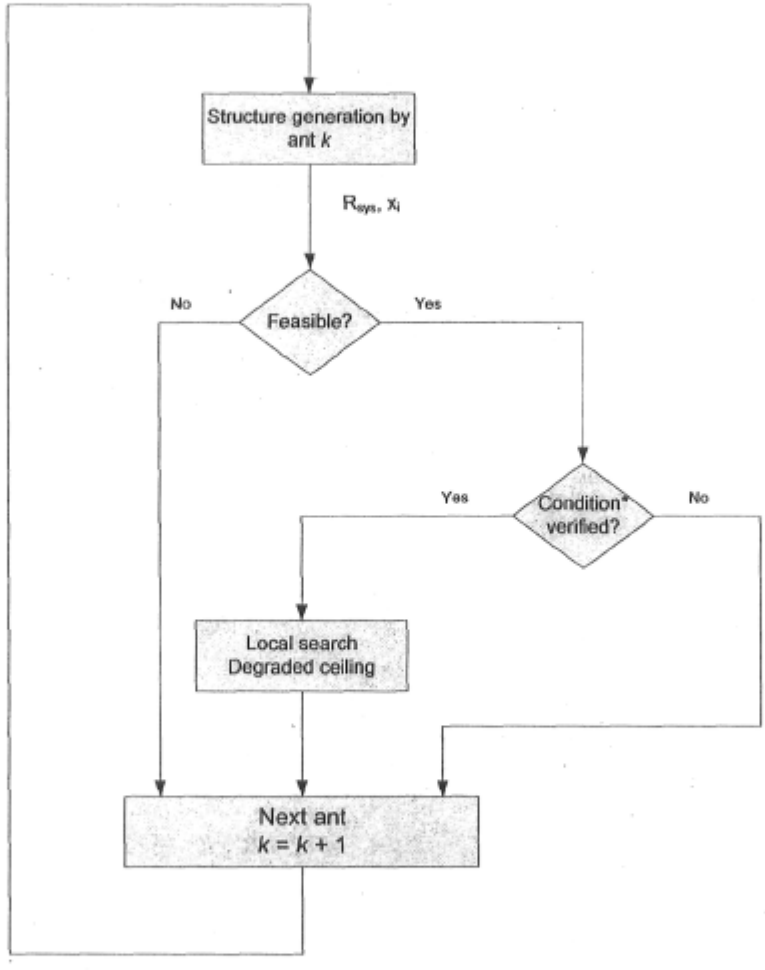

FIG. 2.3 – An overview of an iteration of the hybrid algorithm (ACO/DC).

(*) The condition to be verified is ($R_{sys} \geq R^*$ or $0 \leq \frac{R^*-R_{sys}}{R^*} \leq +l$)

Ants can be guided, in building their tours, by pheromone information and heuristic information. Naturally, an edge with a high amount of pheromone is a very desirable choice. The pheromone updating rules are designed so that they tend to give more

pheromone to edges which should be visited by ants.

In the following we discuss the state transition rules and the global updating rules.

State transition rules

In the ACO algorithm, each ant builds a solution (called a tour) to our problem by repeatedly applying two different state transition rules. At each step of the construction process, ants use : (1) pheromone trails (denoted by τ_{ij}) to select the number of elements connected in parallel and the versions of elements; (2) and a problem-specific heuristic information (denoted by η_{ij}) related to the versions of elements.

An ant positioned on node i (i.e. component i) chooses the total number p_i of elements to be connected in parallel. This choice is done by applying the rule given by :

$$P_{ip_i} = \begin{cases} \frac{(\tau_{ip_i})^{\alpha_1}}{\sum_{k=1}^{p_{\max}}(\tau_{ik})^{\alpha_1}} & \text{if} \quad p_i \in \{1, 2, ..., p_{\max}\} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where α_1 is a parameter that controls the relative weight of the pheromone (τ_{ip_i}). We favour the choice of edges which are weighted with greater amount of pheromone.

When an ant is positioned on node p_i representing the selected number of elements connected in parallel in component i, it has to choose these p_i versions of elements. This choice is done by applying the rule given by :

$$P_{p_ij} = \begin{cases} \frac{(\tau_{p_ij})^{\alpha_2}(\eta_{p_ij})^{\beta}}{\sum_{k=1}^{a_i}(\tau_{p_ik})^{\alpha_2}(\eta_{p_ik})^{\beta}} & \text{if} \quad j \in \{1, 2, ..., a_i\} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where $\alpha_2$and β are respectively parameters that control the relative weight of the pheromone (τ_{p_ij}) and the local heuristic (η_{p_ij}).

We tested different heuristic information and the most efficient was $\eta_{p_ij} = r_{ij}/w_{ij}^3$ where r_{ij} and w_{ij} represent respectively the associated reliability and weight of version j for component i. In equation (2.6) we multiply the pheromone on edges by the corresponding heuristic information. In this way we favour the choice of edges which are weighted with smaller weight and greater reliability and which have a greater amount of pheromone.

Global updating rule

During the construction process, no guarantee is given that an ant will construct a feasible solution which obeys the constraints (2.2) and (2.3). The unfeasibility of solutions is treated in the pheromone update : the amount of pheromone deposited by an ant is set to a high value if the generated solution is feasible and to a low value if it is infeasible. These values are dependent of the solution quality. Infeasibilities can then be handled by assigning penalties proportional to the amount of cost and weight violations. Thus, the trail intensity is updated as follows :

$$\tau_{ij}(\text{new}) = \rho\tau_{ij}(\text{old}) + \Delta\tau_{ij} \tag{2.7}$$

ρ is a coefficient such that (1-ρ) represents the evaporation of trail and is $\Delta\tau_{ij}$:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2.8}$$

where m is the number of ants. Furthermore, $\Delta\tau_{ij}^{k}$ is given by :

$$\Delta\tau_{ij}^{k} = \begin{cases} Q.penalty_k.R_{sys}^{k} & \text{if the edge } (i,j) \text{ is visited by the k}^{th}\text{ant} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where Q is a positive number, R_{sys}^{k} is the system reliability for ant k, and $penalty_k$ is defined as follows :

$$penalty_k = \begin{cases} \left(\dfrac{C}{TC_k}\right)^a & \text{if } (C < TC_k) \\ \left(\dfrac{W}{TW_k}\right)^b & \text{if } (W < TW_k) \\ \left(\dfrac{C}{TC_k}\right)^a \left(\dfrac{W}{TW_k}\right)^b & \text{if } (C < TC_k) \text{ and } (W < TW_k) \end{cases} \tag{2.10}$$

TC_k and TW_k are respectively the total cost and the total weight obtained by ant k. Parameters a and b represent the relative importance of penalties. The detailed ant

system algorithm is presented in figure 2.4. In the next subsection, the degraded ceiling procedure used in our ACO algorithm is explained.

```
1. Initialization
  Set NI = 0                        /*NI: iteration counter*/
  For every combination (i,j)
    /*i-component or number of elements index, j- number of elements or version index*/
    Set an initial value τ_ij(0) = τ_0 and Δ_ij = 0
  End
2. For k = 1 to m                   /*m: number of ants */
/* build the series/parallel structure*/
      For i = 1 to n                /*n: number of component*/
        Choose the number of elements to be connected in parallel using equation (2.5)
        Choose a version with transition probability given by equation (2.6)
      End
      Calculate R^k_sys             /*R^k_sys: system reliability for ant k*/
      If (feasible solution) and (R^k_sys ≥ R* or 0 ≤ (R*-R_sys)/R' ≤ +l)
        Apply the local search procedure (i.e. degraded ceiling.
      End
      Update the best solution R*
  End
3. For k = 1 to m
      Calculate the penalties using equation (2.10)
      Update using equations (2.8) and (2.9)
   End
   For every combination (i,j)
      Update the trail values according to equation (2.7)
      Update the transition probabilities according to equations (2.5) and (2.6)
   End
4. Set NI = NI + 1
   For every combination (i,j), set Δ_ij = 0
5. If (NI < NImax) and (not stagnation behavior)
      Then Goto step 2
   Else Print the best feasible solution and Stop
   End.
```

FIG. 2.4 – ACO/DC algorithm for redundancy allocation problem

2.3.2 The degraded ceiling local search meta-heuristic

The principle of the degraded ceiling meta-heuristic

The performance of algorithms based on the ACO meta-heuristic can be greatly enhanced when coupled to local improvement procedures. A degraded ceiling (DC) based algorithm is included in our ACO approach to improve the solutions obtained by the ants.

The degraded ceiling is a local search meta-heuristic recently introduced in [6] and [7]. Like other local search methods, the degraded ceiling iteratively repeats the repla-

cement of a current solution s by a new one s^*, until some stopping condition has been satisfied. The new solution is selected from a neighbourhood $N(s)$. The mechanism of accepting or rejecting the candidate solution from the neighbourhood is different of other methods. In degraded ceiling approach, the algorithm accepts every solution whose objective function is more or equal (for the maximization problems) to the upper limit L, which is monotonically increased during the search by (ΔL).

The initial value of ceiling (L) is equal to the initial cost function $f(s)$ and only one input parameter ΔL has to be specified. In [6] and [7], the authors applied successfully the degraded ceiling on exam timetabling problem and demonstrated that it outperformed well-known best results found by others meta-heuristics, such as simulated annealing and tabu search. They showed two main properties of the degraded ceiling algorithm :

- The search follows the degrading of the ceiling. Fluctuations are visible only at the beginning, but later, all intermediate solutions lie close to a linear line.
- When a current solution reaches the value where any further improvement is impossible, the search rapidly converges. The search procedure can then be terminated at this moment.

The degraded ceiling algorithm is an extension of the "great deluge" method which was introduced as an alternative to simulated annealing. Degraded ceiling, simulated annealing and "great deluge" algorithms share the characteristic that they may both accept worse candidate solutions than the current one. The difference is in the acceptance criterion of worse solutions. The simulated annealing method accepts configurations which deteriorate the objective function only with a certain probability. The degraded ceiling algorithm incorporates both the worse solution acceptance (of the "great deluge" algorithm) if the solution fitness is less than or equal to some given upper limit L, i.e. $(f(s^*) \geq L)$, and the well-known hill climbing rule $(f(s^*) \geq f(s))$.

Like simulated annealing, the degraded ceiling algorithm may accept worse candidate solutions during its run. The introduction of the dynamic parameter has an important effect on the search. As explained in [6][7], the decreasing of L may be seen as a control process, which drives the search towards a desirable solution. Note finally that degraded ceiling algorithm has the advantage to require only one parameter (ΔL) to be tuned.

The degraded ceiling as a local search in the ACO algorithm for the RAP

Figure 2.5 presents the chart of the degraded ceiling algorithm when applied to the redundancy allocation problem. The limit L is monotonically increased during the search by ΔL, as we deal with a maximisation problem. The degraded ceiling starts with a feasible solution given by the ACO.

In order to adapt the degraded ceiling algorithm to the redundancy allocation problem, it is necessary to specify the type of the adopted neighbourhood. The search proceeds iteratively from one feasible solution to another by moves in the neighbourhood. An initial solution s is given by the set of vectors x_i $(i = 1, 2, \ldots, n)$. The move is made as follows :

Step 1. Choose randomly a number t with $(1 \leq t \leq n)$; initialize a counter $Count = 1$.
Step 2. Choose randomly a component i $(i \in \{1, 2, \ldots, n\})$.Step 3.
Step 3. For the chosen component i, select randomly an element j. Change the element j by another element j' $(j' \neq j)$ randomly selected from the available elements choices for component i.
Step 4. Set $Count = Count + 1$.
Step 5. IF $Count < t$ THEN GO TO Step 2 ELSE Stop.

It is important to remark that our solution encoding and solution construction method is such that two meta-heuristics cooperate in solving the problem. Each meta-heuristic may execute an appropriate and specific task. The ACO task is to determine the number of elements and the versions corresponding to these elements, i.e. to select a structure. The DC task is to explore better neighborhood solutions of this structure. During the DC local search, the quantity of elements in each component remains constant. This allows us to restrict the search space of the DC, and it turned to be very effective as shown later. In fact, the exploration of other search spaces with different quantities of elements in each component is a task already performed by the ants, and it was not necessary to duplicate this task in the DC part of our hybrid algorithm.

In the section 2.4, the proposed hybrid methodology (ACO/DC) is demonstrated on 33 test problems from previous research and compared to the best solutions found in literature. The results indicate that the ACO/DC methodology offers advantages over the other approaches. Before presenting section 2.4, let compare the proposed ACO/DC algorithm to the ant system search described in [27].

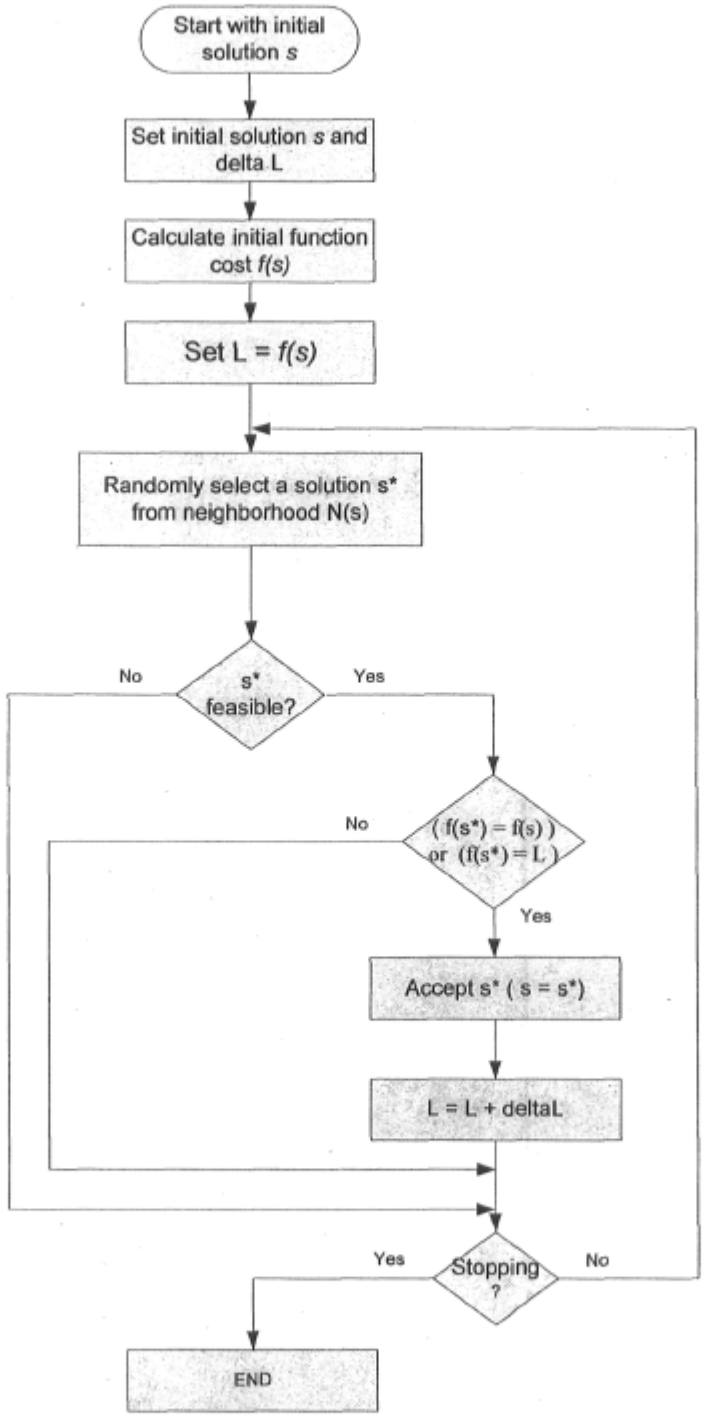

FIG. 2.5 – Degraded ceiling as a local search technique for the RAP solved by ACO

Comparison

The ant system presented in [27] is suitable only for series systems and cannot be used to solve other structure optimization problems, such as the RAP addressed in the present paper. The objective function in the RAP is different from the objective function in the series system problem. The RAP is a combinatorial optimization problem which is much more difficult than the series systems problem. As series system technology choice and series-parallel redundancy allocation are both selection problems, they have naturally intrinsic functional similarities. Nevertheless, the ACO/DC algorithm is conceptually different from the ant system algorithm described in [27].

In general, as explained in [12] and [25], ACO meta-heuristic offers flexibility and is not confined to specific problem types or instances (hence, the "meta"). It is then natural that all ACO algorithms share the same characteristics. In fact, any ACO-based algorithm may correspond to the general schema of figure 2.6

Set all parameters and initialize the pheromone trails
Loop
    Loop
        Each ant applies a state transition rule to construct a solution
Until all ants have built a complete solution
        Evaluate all solutions and record the best feasible one
    Apply a global pheromone updating rule
Until a stopping criterion is reached.

FIG. 2.6 – The general schema of the ACO metaheuristic

The differences between the algorithms based on the ACO meta-heuristic may be either in the way the ants construct their solutions, i.e., in the state transition rule, or in the way the trail intensities are updated, i.e., in the global pheromone updating rule. The newly developed ACO/DC algorithm is different from the ant system search described in [27] in both the state transition rule and the global pheromone updating rule.

Differences in the graph and in the state transition rule

To apply the ACO meta-heuristic to a combinatorial optimization problem, it is convenient to represent this problem as a graph. The graph illustrates simulated ants behavior during the solution process. While in series systems studied in [27] one needs to select for each component only one element among a list of available technologies, in the RAP one must select both the elements and the redundancy levels. In [27], the

graph of the series systems problem is such that the set of nodes is given by components and technologies, and edges connect each component to its available technologies. Only one state transition rule, similar to that of equation (2.6), is used in [27] to choose technologies with higher reliability and smaller cost with higher probability. For the RAP, the designed graph has to be completely different to allow optimal selection of technologies and redundancy levels. Our solution design is based on a two-levels graph (see figure 2.2). That is, to build a solution, an ant must apply two different state transition rules resulting for two different pheromone matrices. In the first graph level, an ant chooses the total number of elements to be connected in parallel by applying the rule given by equation (2.5). In the second graph level, the ant has to choose versions of elements by applying the rule given by equation (2.6).

Differences in the global updating rule

The terms of the global updating rules are different and the penalty factor, introduced to deal with constraints, is different since the series problem in [27] deals only with budget constraint, while the RAP considers budget and weight constraints.

Difference in the local search procedure

It is well known that the performance of ACO algorithms can be greatly enhanced when coupled to local search procedures [12]. Following this, local search algorithms are used in [27] and in the present paper to improve the quality of the solutions obtained by each ant.

In [27], two local improvement algorithms are included. The first algorithm uses the remaining budget (the amount not used by the ant) to improve the solution. This algorithm improves the initial solution by using this remaining budget to exchange some actual technologies by more reliable other technologies. The second algorithm proceeds to change in turn each pair of chosen technologies by another pair.

The local search used in the present paper is based on a recent meta-heuristic (i.e., the degraded ceiling or DC), which can be seen as a variant of the well-known simulated annealing algorithm. Our ACO/DC approach corresponds therefore to a hybridization of two meta-heuristics. Using two meta-heuristics was necessary for the RAP because it is much more difficult than the series problem in [27]. The study conducted in section 2.4 will show that hybridization of ACO and DC meta-heuristics leads to solutions which are all better than or are in par with the well-known best solutions for the RAP.

2.4 Test problems and results

The test problems, used to evaluate the performance of the ACO/DC methodology for the RAP, were originally proposed by Fyffe et al. in [16] and modified by Nakagawa and Miyazaki in [28]. Fyffe et al. [16] specified constraint limits of 130 units of system cost, 170 units of system weight and $k_i = 1$ (i.e., 1-out-of-n :G). Nakagawa and Miyazaki [28] developed 33 variations of the original problem, where the cost constraint C is set to 130 units and the weight constraint W varies from 159 units to 191 units. The element cost, weight and reliability values, as originally presented in [16], are reproduced in Table 2.1. The system is designed with 14 components. For each component, there are three or four element choices.

Component	Element choices											
i	r_{i1}	c_{i1}	w_{i1}	r_{i2}	c_{i2}	w_{i2}	r_{i3}	c_{i3}	w_{i3}	r_{i4}	c_{i4}	w_{i4}
1	0.95	2	5	0.93	1	4	0.91	2	2	0.90	1	3
2	0.95	2	8	0.94	1	10	0.93	1	9			
3	0.92	4	4	0.90	3	5	0.87	1	6	0.85	2	7
4	0.87	4	6	0.85	5	4	0.83	3	5			
5	0.95	3	5	0.94	2	4	0.93	2	3			
6	0.99	3	5	0.98	3	4	0.97	2	5	0.96	2	4
7	0.94	5	9	0.92	4	8	0.91	4	7			
8	0.91	6	6	0.90	5	7	0.81	3	4			
9	0.99	3	9	0.97	2	8	0.96	4	7	0.91	3	8
10	0.90	5	6	0.85	4	5	0.83	4	6			
11	0.96	5	6	0.95	4	6	0.94	3	5			
12	0.90	5	7	0.85	4	6	0.82	3	5	0.79	2	4
13	0.99	3	5	0.98	2	5	0.97	2	6			
14	0.99	6	9	0.95	5	6	0.92	4	7	0.90	4	6

TAB. 2.1 – Component parameters for testing problem

Earlier optimization approaches to the problem (e.g., [16] and [[28]), required that only identical elements be placed in redundancy. Such approaches determined optimal solutions through dynamic programming and IP models, but only a restricted set of solutions was considered due to computational or formulation limitations of exact solution methods. Nevertheless, for the ACO/DC approach, as in [9], [22] and [25], different types are allowed to reside in parallel. In [9], Coit and Smith first solved the RAP with a genetic algorithm without restricting the search space. More recently, Kulturel-Konak et al. solved this problem in [22] with a tabu search algorithm, while Liang and Smith

[25] used an ant colony optimization approach improved by a local search. Because the heuristic benchmarks for the RAP where elements mixing is allowed are the methods in [9], [22] and [25], there are chosen for comparison. Our approach will be compared also with [28] and [20]. By comparing the proposed ACO/DC methodology to all the above-mentioned papers (e.g., [9], [22], [25], [28] and [20]), we compare it to the best-known solutions found in literature at the best of our knowledge.

In meta-heuristics such as ACO, simulated annealing and degraded ceiling, it is necessary to tune a number of parameters to have good performance. The user-specified parameters of the ACO/DC algorithm were varied to establish the values most beneficial to the optimization process. Following the tuning procedure used in [12]-[14] and [27], we tested various values for each parameter, while keeping the others constant. Based on these initial experiments the values found to be most appropriate are :

$\alpha_1 = 0.1$, $\alpha_2 = 0.5$, $\beta = 1, Q = 0.01, \rho = 0.9, a = 1, b = 10, \tau_0 = 1, \Delta L = 0.0001$ and $l = 0.01$.

These parameters values are used for all test problems. 50 ants are used in each iteration. When combined to the degraded ceiling algorithm, ACO converges quickly to optimal or near optimal solutions. Note that the degraded ceiling is called only if the obtained solutions are very good. For the considered problem instances, the maximum number of iterations needed does not exceed 300 iterations.

The configuration of the best solution, and its system reliability, cost and weight for each of the 33 instances, are shown in Table 2.2. In this table, the average and the maximum of the best solution found among 5 runs are also presented as well as the standard deviations of the 5 final solutions. The standard deviation is an important measure of algorithm robustness. From Table 2.2, for each case, the standard deviation is very low. This implies that the proposed approach is robust and credible. Table 2.3 presents the corresponding results from [9][20][22][25][28]. As the results in Table 2.3 indicate, ACO/DC generally out-performed the existing approaches.

Comparing the results obtained by our approach with those of previous works [9][20][22][25][28], Table 2.3 shows that :

1. The solutions found by our algorithm are all better than those of Hsieh [20].
2. In 31 of the 33 test cases, the solutions found by our algorithm are better than those of Nakagawa and Miyazaki [28] while the rest (i.e., 2 cases) are as good as those they found.
3. Cases 22 to 29 and 31 to 32 are as good as those found by the genetic algorithm

of Coit and Smith [9] while the rest (i.e., 24 instances) are all better than those they found.

4. In 6 of the 33 test cases, the ACO/DC outperformed the tabu search algorithm of Kulturel-Konak et al. [22] while it was very close but at a lower reliability in only 2 instances.
5. In 9 of the 33 test cases, the solutions found by our algorithm are better than those of Liang and Smith [25] while the rest are as good as those they found.

The hybridization suggested in this chapter showed very good numerical results compared to approaches presented in the literature. However, the proposed approach is time consuming (between 20 and 30 minutes). To overcome this disavantage, an improvement of this hybridization is proposed in the next section.

2.5 Improvement of the ACO/DC algorithm

The major improvements made on the hybrid ACO/DC algorithm are as follow :

- The ACO and DC algorithms will have different tasks for solutions : The ACO algorithm task consists in choosing *only* the number of elements in parallel on each component, while the DC algorithm task is to find the best versions for the elements. Figure 2.7 shows how a solution will be constructed and improved by the improved ACO/DC algorithm.
- The convergence time of the DC algorithm is greatly decreased : the parameter ΔL is fixed at 0.0003 and the maximum allowed number of iterations is 2000.
- The number of ants is fixed to 5 (instead of 50 in the previous version).
- Based on the work of [14], the ACO algorithm presented previously is modified as follows : An ant positioned on node i (i.e. component i) chooses the total number p_i of elements to be connected in parallel according to the following rule :

$$p_i = \begin{cases} \arg\max_{k\in\Gamma_i} \{(\tau_{ik})^\alpha\} & \text{if } q \leq q_0 \\ J & \text{Otherwise} \end{cases} \tag{2.11}$$

where Γ_i is the set of nodes representing the possible numbers of elements which can be connected in parallele (i.e. $\Gamma_i = \{1, \ldots, p_{\max}\}$) ; and J is a random variable selected according to the probability distribution given by :

$$p_{ip_i} = \begin{cases} \frac{(\tau_{ip_i})^\alpha}{\sum_{k=1}^{p_{\max}} (\tau_{ik})^\alpha} & \text{if} \quad p_i \in \{1, 2, .., p_{\max}\} \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

In the above equations (2.11) and (2.12), α is a parameter that control the relative weight of the pheromone (τ_{ij}). The variable q is a random number uniformly distributed in $[0, 1]$; and q_0 is a parameter ($0 \leq q_0 \leq 1$) which determines the relative importance of exploitation versus exploration.

Global updating rule

The pheromone trails are updated once all ants have chosen the number of elements to be connected in parallel for each component and the obtained solutions are improved by the DC algorithm. Only the globally best ant (i.e., the ant which constructed the best design solution from the beginning of the trial) is allowed to deposit pheromone. A quantity of pheromone $\Delta\tau_{ij}$ is deposited on each edge that the best ant has used. The quantity $\Delta\tau_{ij}$ is given by the reliability R_{best} of the feasible solution constructed by the best ant. Therefore, the global updating rule is :

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{2.13}$$

where $0 < \rho < 1$ is the pheromone decay parameter representing the evaporation of trail.
Global updating is intended to allocate a greater amount of pheromone to greater design solution. Equation (2.13) dictates that only those edges belonging to the globally best solution will receive reinforcement.

Local updating rule

While building a solution of the problem, ants visit edges on the graph G, and change their pheromone level by applying the following local updating rule :

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\tau_0 \tag{2.14}$$

where τ_0 is the initial value of trail intensities.
The application of the local updating rule, while edges are visited by ants, has the effect of lowering the pheromone on visited edges. This favours the exploration of edges not yet visited, since the visited edges will be chosen with a lower probability by the other ants in the remaining steps of an iteration of the algorithm.

- The value of the objective function is not penalized and the infeasible solutions will be simply rejected.
- When the number of elements in parallel is chosen by an ant, the feasible initial solution used by DC algorithm is built by affecting the versions having the minimal weight.

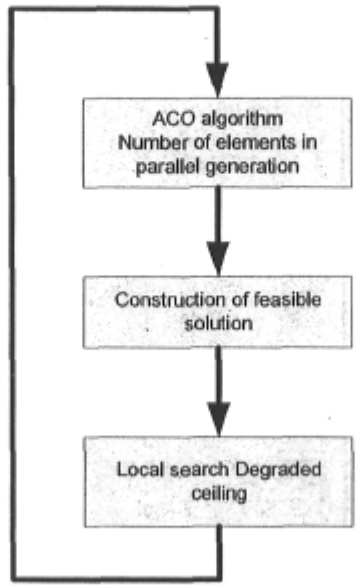

FIG. 2.7 – Solution construction by the improved ACO/DC algorithm

The performance of the new hybrid ACO/DC has been evaluated on the 33 test problems with the following parameters : $\rho = 0.9$, $\alpha = 0.5$, $q_0 = 0.97$, $\tau_0 = 0.2$. Table 2.4 presents a comparison between the new structure of the algorithm with that presented previously. The average computing time of the improved hybrid algorithm is approximately 650 seconds (about 11 minutes). Table 2.4 shows that the same results are obtained by the new hybridization for 31 cases and in 2 cases (instances 2 and 9), the results are improved.

2.6 Conclusions

Both the degraded ceiling and the ant colony algorithms are meta-heuristics. Our contribution is based on the ACO/DC hybridization and very good results are obtained. The RAP is one of the most difficult combinatorial optimization problems inherent to optimal design of reliable systems. We believe and we show that two efficient meta-heuristics have to cooperate in order to solve efficiently this problem, namely the ACO and the DC meta-heuristics.

The study conducted in this article shows that hybridization of meta-heuristics is a very promising methodology for NP-hard reliability design problems. We intend to apply the ACO/DC (hybrid) methodology developed in this work to the redundancy optimization for multi-state systems.

In the proposed ACO/DC approach, it has been necessary to tune many parameters. The development of a systematic procedure for parameters fine-tuning could be investigated in further studies. For a pure ACO, a good starting point for parameter tuning should be the use of parameter values that were verified to be good when applying the ACO algorithm to similar problems. This was not possible in our case because the proposed hybrid ACO/DC algorithm is new. An alternative to time-consuming personal involvement in the tuning task is to develop fully automatic procedures for parameter settings.

No	W	$R_{\max}$	R_{av}	Std.dev.	Cost	Weight	Solution
1	191	0.9868110	0.9866127	0.000390	130	191	333,11,111,2222,333,22,333,3333,12,112,11,4444,22,12
2	190	0.9863162	0.9860909	0.000519	130	190	333,11,111,2222,333,22,333,3333,12,112,31,4444,11,12
3	189	0.9859216	0.9856668	0.000173	130	189	333,11,111,2222,333,22,333,3333,31,112,31,4444,22,21
4	188	0.9853782	0.9852074	0.000240	130	188	333,11,111,2222,333,22,333,3333,31,122,13,4444,12,21
5	187	0.9846880	0.9846801	0.000010	130	187	333,11,111,2222,333,22,333,3333,23,122,13,4444,11,21
6	186	0.9841755	0.9841635	0.000016	129	186	333,11,111,222,333,22,333,3333,31,112,11,4444,11,12
7	185	0.9835048	0.9834637	0.000037	130	185	333,11,111,2222,333,22,333,3333,13,221,31,4444,11,22
8	184	0.9829940	0.9828911	0.000230	130	184	333,11,111,222,333,22,333,3333,33,211,11,4444,11,12
9	183	0.9822259	0.9820781	0.000184	130	183	333,11,111,2222,333,22,333,3333,33,212,31,4444,12,22
10	182	0.9815183	0.9814884	0.000027	130	182	333,11,111,222,333,22,333,3333,33,111,11,4444,11,22
11	181	0.9810270	0.9809119	0.000257	129	181	333,11,111,222,333,22,333,3333,33,121,11,4444,11,22
12	180	0.9802901	0.9802901	0	128	180	333,11,111,222,333,22,333,3333,33,212,11,4444,11,22
13	179	0.9795047	0.9795047	0	126	179	333,11,111,222,333,22,333,3333,33,221,13,4444,11,22
14	178	0.9784002	0.9783882	0.000010	125	178	333,11,111,222,333,22,333,3333,33,222,31,4444,11,22
15	177	0.9775963	0.9775717	0.000054	126	177	333,11,111,222,333,22,333,331,33,122,31,4444,11,22
16	176	0.9766904	0.9766904	0	124	176	333,11,111,222,333,22,11,3333,33,221,31,4444,11,22
17	175	0.9757079	0.9757079	0	125	175	333,11,111,222,333,22,31,3333,33,212,11,4444,11,22
18	174	0.9749260	0.9748983	0.000061	123	174	333,11,111,222,333,22,31,3333,33,212,13,4444,11,22
19	173	0.9738268	0.9738228	0.000008	122	173	333,11,111,222,333,22,13,3333,33,222,31,4444,11,22
20	172	0.9730266	0.9730266	0	123	172	333,11,111,222,333,22,31,313,33,221,31,4444,11,22
21	171	0.9719294	0.9719294	0	122	171	333,11,111,222,333,22,31,133,33,222,13,4444,11,22
22	170	0.9707603	0.9704068	0.000603	122	170	333,11,111,222,333,22,33,313,33,221,31,4444,11,22
23	169	0.9692910	0.9690508	0.000537	121	169	333,11,111,222,333,22,33,313,33,222,31,4444,11,22
24	168	0.9681250	0.9679998	0.000280	119	168	333,11,111,222,333,22,33,133,33,222,33,4444,11,22
25	167	0.9663351	0.9663088	0.000058	118	167	333,11,111,222,33,22,31,331,33,222,33,4444,11,22
26	166	0.9650416	0.9648934	0.000331	116	166	333,11,11,222,333,22,33,3333,33,222,33,4444,11,22
27	165	0.9637118	0.9634730	0.000534	117	165	333,11,111,222,33,22,33,133,33,222,33,4444,11,22
28	164	0.9624218	0.9622973	0.000278	115	164	333,11,11,222,333,22,33,313,33,222,33,4444,11,22
29	163	0.9606424	0.9606424	0	114	163	333,11,11,222,33,22,31,133,33,222,33,4444,11,22
30	162	0.9591883	0.9590875	0.000138	115	162	333,11,11,222,33,22,33,313,33,222,13,4444,11,22
31	161	0.9580345	0.9577863	0.000555	113	161	333,11,11,222,33,22,33,133,33,222,33,4444,11,22
32	160	0.9557144	0.9557144	0	112	160	333,11,11,222,33,22,33,333,33,222,13,4444,11,22
33	159	0.9545648	0.9541790	0.000528	110	159	333,11,11,222,33,22,33,333,33,222,33,4444,11,22

TAB. 2.2 – Configuration, reliability, cost, and weight obtained by the ACO/DC algorithm (5 trials)

No	W	Nakagawa and Miyazaki (1981)	Coit and Smith (1996)	Hsieh (2002)	Kulturel et al. (2003)	Liang and Smith (2004)	ACO/DC
		Reliability	Reliability	Reliability	Reliability	Reliability	Reliability
1	191	0.9864	0.98675	0.98671	0.986811	0.986745	0.986811
2	190	0.9854	0.98603	0.98631	0.986416	0.985905	0.986316
3	189	0.9850	0.98556	0.98572	0.985922	0.985773	0.985922
4	188	0.9847	0.98503	0.98503	0.985378	0.985329	0.985378
5	187	0.9840	0.98429	0.98415	0.984688	0.984688	0.984688
6	186	0.9831	0.98362	0.98387	0.984176	0.983801	0.984176
7	185	0.9829	0.98311	0.98338	0.983505	0.983505	0.983505
8	184	0.9822	0.98239	0.98220	0.982994	0.982994	0.982994
9	183	0.9815	0.98190	0.98146	0.982256	0.982206	0.982225
10	182	0.9815	0.98102	0.97969	0.981518	0.981468	0.981518
11	181	0.9800	0.98006	0.97928	0.981027	0.980681	0.981027
12	180	0.9796	0.97942	0.97832	0.980290	0.980290	0.980290
13	179	0.9792	0.97906	0.97805	0.979505	0.979505	0.979505
14	178	0.9772	0.97810	0.97687	0.978400	0.978400	0.978400
15	177	0.9772	0.97715	0.97540	0.977474	0.977596	0.977596
16	176	0.9764	0.97642	0.97497	0.976690	0.976494	0.976690
17	175	0.9744	0.97552	0.97350	0.975708	0.975708	0.975708
18	174	0.9744	0.97435	0.97232	0.974788	0.974926	0.974926
19	173	0.9723	0.97362	0.97053	0.973827	0.973827	0.973827
20	172	0.9720	0.97266	0.96923	0.973027	0.973027	0.973027
21	171	0.9700	0.97186	0.96789	0.971929	0.971929	0.971929
22	170	0.9700	0.97076	0.96677	0.970760	0.970760	0.970760
23	169	0.9675	0.96922	0.96561	0.969291	0.969291	0.969291
24	168	0.9666	0.96813	0.96415	0.968125	0.968125	0.968125
25	167	0.9656	0.96634	0.96299	0.966335	0.966335	0.966335
26	166	0.9646	0.96504	0.96121	0.965042	0.965042	0.965042
27	165	0.9621	0.96371	0.95992	0.963712	0.963712	0.963712
28	164	0.9609	0.96242	0.95860	0.962422	0.962422	0.962422
29	163	0.9602	0.96064	0.95731	0.959980	0.960642	0.960642
30	162	0.9589	0.95912	0.95554	0.958205	0.959188	0.959188
31	161	0.9565	0.95803	0.95410	0.956922	0.958034	0.958034
32	160	0.9546	0.95567	0.95295	0.955604	0.955714	0.955714
33	159	0.9546	0.95432	0.95080	0.954325	0.954564	0.954564

TAB. 2.3 – Comparison of the ACO/DC and the solutions found in literature

No	W	ACO/DC Best solution		Improved ACO/DC		
		Reliability		$R_{\max}$	R_{av}	Std. dev
1	191	0.986811		0.986811	0.98654	0.000380
2	190	0.986316		**0.986416**	0.98620	0.000381
3	189	0.985922		0.985922	0.98545	0.000514
4	188	0.985378		0.985378	0.985087	0.000441
5	187	0.984688		0.984688	0.984688	0
6	186	0.984176		0.984176	0.984110	0.000173
7	185	0.983505		0.983505	0.983434	0.000147
8	184	0.982994		0.982994	0.982937	0.000113
9	183	0.982225		**0.982255**	0.982237	0.000153
10	182	0.981518		0.981518	0.981518	0
11	181	0.981027		0.981027	0.980869	0.000332
12	180	0.980290		0.980290	0.980211	0.000249
13	179	0.979505		0.979505	0.979207	0.000934
14	178	0.978400		0.978400	0.978400	0.000117
15	177	0.977596		0.977596	0.977510	0.000592
16	176	0.976690		0.976690	0.976007	0.002024
17	175	0.975708		0.975708	0.975333	0.000790
18	174	0.974926		0.974926	0.974697	0.000724
19	173	0.973827		0.973827	0.973627	0.000629
20	172	0.973027		0.973027	0.973027	0
21	171	0.971929		0.971929	0.971929	0
22	170	0.970760		0.970760	0.970760	0
23	169	0.969291		0.969291	0.968930	0.000580
24	168	0.968125		0.968125	0.968062	0.000198
25	167	0.966335		0.966335	0.962586	0.011715
26	166	0.965042		0.965042	0.964957	0.000089
27	165	0.963712		0.963712	0.963522	0.000381
28	164	0.962422		0.962422	0.962422	0
29	163	0.960642		0.960642	0.960642	0
30	162	0.959188		0.959188	0.958715	0.001496
31	161	0.958034		0.958034	0.958034	0
32	160	0.955714		0.955714	0.955714	0
33	159	0.954564		0.954564	0.954564	0

TAB. 2.4 – Results obtained whith improved ACO/DC

Bibliographie

[1] Bauer A, Bullnheimer B, Hartl RF and Strauss C. *Minimizing total tardiness on a single machine using ant colony optimization.* Central Eur. J. Oper. Res., 2000 ;8(2), 125–41.

[2] Bellman RE and Dreyfus E. *Dynamic programming and reliability of multicomponent devices.* Operations Research, 1958,6, 200-06.

[3] Besteo MD, Stützle T and Dorigo M. *Ant colony optimization for the total weighted tardiness problem.* Proc. 6th Int. Conf. Parallel Problem Solving From Nature (PPSN VI), Berlin, 2000, 611–20.

[4] Bulfin RL and Liu CY. *Optimal allocation of redundant components for large systems.* IEEE Transactions on Reliability, 1985, 34, 241-247.

[5] Bullnheimer B, Hartl RF and Strauss C. *Applying the Ant System to the vehicle Routing problem.* 2nd Metaheuristics International Conference (MIC-97), Sophia-Antipolis, France, 1997, 21-24.

[6] Burke EK, Bykov Y, Newall JP and Petrovic S. *A New Local Search Approach with Execution Time as an Input Parameter.* Computer Science Technical Report No. NOTTCS-TR-2002-3. School of Computer Science and Information Technology. University of Nottingham.

[7] Burke EK, Bykov Y, Newall JP and Petrovic S. *A time-predefined local search approach to exam timetabling problems.* IIE Transactions, 2004, 36(6), 509-28.

[8] Chern MS. *On the computational complexity of reliability redundancy allocation in a series system.* Operations Research Letter 1992 ;11 :309–15.

[9] Coit DW and Smith AE. *Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm.* IEEE Transactions on Reliability, 1996, 45(2), 254-60.

[10] Costa D and Hertz A. *Ants can color graphs.* J. Oper. Res. Soc., 1997, 48, 295–305.

[11] Dorigo M. *Optimization, Learning and Natural Algorithms.* Ph.D Thesis, Politecnico di Milano, Italy, 1992.

[12] Dorigo M, Maniezzo V and Colorni A. *The Ant System : Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man and Cybernetics- Part B, 1996, 26(1), 1-13.

[13] Dorigo M and Gambardella LM. *Ant colonies for the traveling salesman problem.* BioSystems, 1997, 43, 73–81.

[14] Dorigo M and Gambardella LM. *Ant colony system : a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation, 1997, 1(1), 53–66.

[15] Di Caro G and Dorigo M. *Mobile Agents for Adaptive Routing.* Proceedings for the 31st Hawaii International Conference on System Sciences, Big Island of Hawaii, January 6-9, 1998, 74-83.

[16] Fyffe DE, Hines WW and Lee NK. *System Reliability Allocation And a Computational Algorithm.* IEEE Transactions on Reliability, 1968, R-17(2), 64-69.

[17] Gambardella LM, Taillard E and Dorigo M. *Ant Colonies for the Quadratic Assignment Problem.* Journal of the Operational Research Society, 1999, 50, 167-76.

[18] Gen M, Ida K, Tsujimura Y and Kim CE. *Large scale 0-1 fuzzy goal programming and its application to reliability optimization problem.* Computers & Industrial Engineering, 1993, 24, 539-49.

[19] Ghare M and Taylor RE. *Optimal redundancy for reliability in series system.* Operations research, 1969, 17, 838-47.

[20] Hsieh YC. *A linear approximation for redundant reliability problems with multiple components choices.* Computers and Industrial Engineering, 2002, 44, 91-03.

[21] Glover F. *Future paths for integer programming and links to artificial intelligence.* Computers & Operations Research, 1986, 13, 533-49.

[22] Kulturel-Konak S, Smith AE and Coit D.W. *Efficiently solving the redundancy allocation problem using tabu search.* IIE transactions, 2003, 35, 515-26.

[23] Kuo W and Prasad VR. *An annotated overview of system-reliability optimization.* IEEE Transactions on Reliability, 2000, 49(2), 176–87.

[24] Levitin G, Lisnianski A, Ben-Haim H and Elmakis D. *Redundancy optimization for series-parallel multi-state systems.* IEEE Transactions on Reliability, 1998, 47(2), 165-72.

[25] Liang Y.C and Smith A.E. *An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP).* IEEE transactions on Reliability, 2004, 53 (3), 417-23.

[26] Misra KB and Sharma U. *An Efficient Algorithm to Solve Integer-Programming Problems Arising in System-Reliability Design.* IEEE Transactions on Reliability, 1991, 40(1), 81-91.

[27] Nahas N and Nourelfath M. *Ant system for reliability optimization of a series system with multiple-choice and budget constraints.* Reliab Eng Syst Saf , 2005, 87(1), 1–12.

[28] Nakagawa Y and Miyazaki S. *Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints.* IEEE Transactions on Reliability, 1981, R-30(2), 175-80.

[29] Nourelfath M, Nahas N and Ait-Kadi D. *Optimal design of series production lines with unreliable machines and finite buffers.* Journal of Quality in Maintenance Engineering, 2005, 11(2), 121-38.

[30] Painton L and Campbell J. *Genetic Algorithms in Optimization of System Reliability.* IEEE Transactions on Reliability, 1995, 44(2), 172-78.

[31] Schoofs L and Naudts B. *Ant colonies are good at solving constraint satisfaction problems.* Proc. 2000 Congress on Evolutionary Computation, San Diego, CA, July 2000, 1190–195.

[32] Tillman FA, Hwang CL and Kuo W. *Optimization techniques for system reliability with redundancy- a review.* IEEE Transaction on Reliability, 1977, R-26(3), 147-155.

[33] Tillman FA, Hwang C-L and Kuo W. *Determining Component Reliability and Redundancy for Optimum System Reliability.* IEEE Transactions on Reliability, 1977, R-26(3), 162-165.

[34] Yalaoui A, Châtelet E and Chu C. *A New Dynamic Programming Method for Reliability and Redundancy Allocation in a Parallel-Series System.* IEEE transactions on Reliability, 2005, 54 (2), 254-261.

[35] Yokota T, Gen M and Ida K. *System reliability of optimization problems with several failure modes by genetic algorithm.* Japanese Journal of Fuzzy Theory and Systems, 1995, 7(1), 117–135.

[36] Yokota T, Gen M and Li YX. *Genetic algorithm for nonlinear mixed-integer programming and its applications.* Computers and Industrial Engineering, 1996, 30(4), 905–917.

[37] Wagner IA and Bruckstein AM. *Hamiltonian(t)—an ant inspired heuristic for recognizing Hamiltonian graphs.* Proc. 1999 Congress on Evolutionary Computation, Washington, D.C., July 1999, 1465–469.

Chapitre 3

A new approach for buffer allocation in unreliable production lines

Résumé

Ce chapitre traite le problème d'allocation optimale des stocks tampons dans les lignes de production non fiables en appliquant une nouvelle approche basée sur l'algorithme du grand déluge etendu. Dans cette étude, deux types de lignes ont été traitées : les lignes homogènes et les lignes non-homogènes. Afin de prouver l'efficacité de notre approche, une étude comparative a été réalisée avec le recuit simulé qui donne les meilleurs résultats publiés dans la littérature. Les résultats montrent que l'algorithme proposé est très compétitif tant au niveau du temps de convergence qu'au niveau de la qualité des solutions.

3.1 Abstract

In this article, we describe a new local search approach for solving the buffer allocation problem in unreliable production lines. In this problem, we need to determine near optimal buffer allocation plans in large production lines and the objective is to maximize the average throughput. An analytical decomposition-type approximation is used to estimate the production line throughput. The proposed approach allows the allocation plan to be calculated subject to a given amount of total buffers slots in a computationally efficient way.

3.2 Introduction

A production line consists of machines connected in series and separated by buffers. Each part is required to be processed on each machine during a fixed amount of time called processing time. A production line for which the processing times are equal at all machines will be called a homogeneous (or balanced) line. In a non-homogeneous (or non-balanced) line, machines may take different lengths of time performing operations on parts. The optimal buffer allocation is an important research issue in the design of homogeneous and non-homogeneous lines. It consists of devising an allocation plan for distributing a certain amount of buffer space among the intermediate buffers of a production line. The efficiency of the system can be then improved by providing optimally the sizes of the buffers between the machines. To deal with this problem, two different tools are needed : an evaluation tool and an optimization tool. The first one allows the determination of the performance measure of the production line (e.g. throughput) which has to be optimized. The second tool is the optimization method that tries to find an optimal or near optimal solution for the problem.

Several authors have been working on the buffer allocation problem (BAP). Buzacott [3] solved this problem by using Markov chain models. So [22] studied the optimal allocation of buffer units with the objective of minimizing work-in-process in production lines. Most of the studies were made on short reliable production lines (e.g. [10][11]). Recently, some authors [19][20][21] proposed an application of the simulated annealing on the buffer allocation problem. They proved the efficiency of this method on large reliable production lines. Singh and MacGregor Smith [18] and Papadopoulos et al. [16] present a classification of the research work in this domain, and they highlight that simulated annealing is a computationally efficient way to solve the BAP.

The simulated annealing (SA) exploits the analogy between the way in which a metal cools and freezes into a minimum energy crystalline energy and the search for a minimum in a more general energy. The connection between this algorithm and mathematical minimization was first noted by Pincus [17] but it was Kirckpatrick et al. [13] who proposed that it form the basis of optimization technique for combinatorial problems. An overview of its different applications is given in [14]. It is however well-known that a common drawback of local search metaheuristiçs such as simulated annealing in solving combinatorial optimization problems is the necessity to set a number of uncertain parameters. This makes the algorithm problem-dependant and significantly increases the total time of solving the problem. On the other hand, the methods, without any parameters (such as hill-climbing), usually produces results of inferior quality.

In this article, we use a variant of a local search metaheuristic, called the degraded ceiling approach, for solving the BAP. This approach was recently introduced in [1], where it has been tested on real-world university examination timetabling problems and the experiments have confirmed its high effectiveness. The degraded ceiling approach developed in this paper requires the setting of only one parameter. This parameter can be interpreted as search time. Generally, a longer search provides a better result, as long as one can intelligently stop the approach from converging too early. Therefore, a user can choose a balance between processing time and the quality of the solution. The efficiency of the proposed optimization method is shown by comparing it to the simulated annealing approach, which has been proven to be very effective in [19][20][21]. For example, in [20], the authors show that both simulated annealing and genetic algorithm methods can be used for optimizing large line configurations with simulated annealing producing more optimal configurations and the genetic algorithm approach leading in performance. As an evaluation tool, for the homogeneous lines, the two algorithms (i.e., the degraded ceiling and the simulated annealing) use the DDX algorithm developed by [5][4]. The DDX method uses the decomposition method of Gershwin [9]. For the non homogeneous lines the efficient and fast ADDX algorithm developed by Burman [2] is implemented.

The remainder of this paper is organized as follows. In section 3.3, the optimal buffer allocation problem is formulated. Section 3.4 describes the proposed degraded ceiling approach and a brief description of the simulated annealing algorithm. In section 3.5, we provide numerical results obtained from the degraded ceiling approach and the simulated annealing algorithm. Conclusions are given in section 3.6.

3.3 The buffer allocation problem

In this paper, we address the buffer allocation problem in the design of production lines that comprise a sequence of machines in series. The machines may have different processing rates and between each pair of machines there is an intermediate location for storage. Each part enters from the first machine, passes in order from all machines and the intermediate buffers and exits the line from the last machine. The machines are considered unreliable. Once it has failed, the machine takes some time to be repaired. We assume that the first machine is never starved and the last machine is never blocked. Assume that there are n machines and $(n-1)$ buffers in a production line. The objective is to maximize the line throughput, subject to a given total buffer space. That is,

$$\text{Maximize} \quad f(B_1, B_2, ..., B_{n-1}) \tag{3.1}$$

$$\text{Subject to} \quad \sum_{i=1}^{n-1} B_i = K \tag{3.2}$$

$$B_i \geq 0 \qquad i = 1, ..., n-1 \tag{3.3}$$

In equations (3.1), (3.2) and (3.3), the quantity B_i represents the feasible buffer allocation to the ith allocation zone. $f(B_1, B_2, \ldots, B_{n-1})$ is the throughput of the production line to be maximized. K is the total buffer capacity. The number of feasible allocations of K buffer slots among the $(n-1)$ intermediate buffer locations increases dramatically with K and n and is given by equation (3.4) :

$$\begin{pmatrix} K+n-2 \\ n-2 \end{pmatrix} = \frac{(K+1)(K+2)...(K+n-2)}{(n-2)!} \tag{3.4}$$

To solve the above buffer allocation problem (BAP), we utilize the decomposition method, as an evaluation tool, to determine the mean throughput of the lines. To find the buffer allocation that maximizes the throughput of the line, we utilize the degraded ceiling method specifically adapted for solving this problem. This solution methodology is detailed in section 3.4.

3.4 Solution methodology

3.4.1 The evaluation method

The decomposition techniques developed in the literature are efficient in estimating performance characteristics of series production lines. In these techniques it is necessary for each machine to be described by three parameters : failure rate, repair rate and processing rate. Denote the failure rate, the repair rate and the processing rate of the machine i by μ_i, λ_i, and P_i, respectively. We approximate the flow of discrete parts in the production line by a continuous flow. Therefore, the quantity of material in each buffer location i at any time t is a real number taking its value in the interval $[0, B_i]$. In [4], the authors developed decomposition equations for the continuous material model and an efficient algorithm (DDX algorithm) for the homogeneous lines (i.e. the processing times are equal at all machines) to determine the parameters of every two-machines line, such

that all the two-machines lines have the same efficiency, and thus the conservation of flow is maintained. For more details about DDX algorithm, the reader is referred to [4]. In [2], the author extended the decomposition method to the case of non-homogeneous lines and developed an efficient and fast algorithm called ADDX (Accelerated-DDX). In this paper, the ADDX algorithm was implemented to evaluate the mean throughput in the case of non-homogeneous lines.

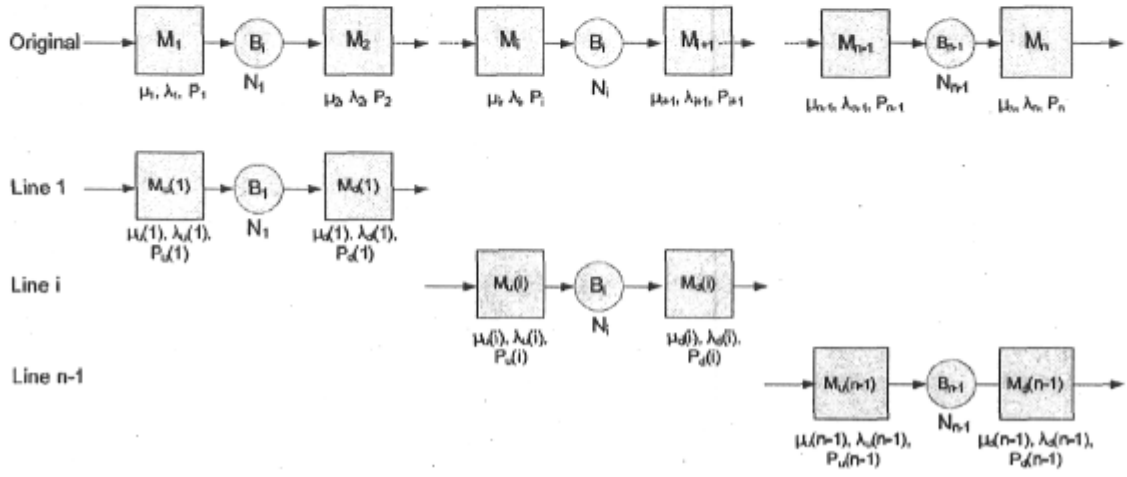

FIG. 3.1 – Decomposition method

3.4.2 The degraded ceiling approach

The degraded ceiling is a local search technique introduced in [1]. Like other local search methods, the degraded ceiling iteratively repeats the replacement of a current solution s by a new one s^*, until some stopping condition has been satisfied. The new solution is selected from a neighborhood $N(s)$. The mechanism of accepting or rejecting the candidate solution from the neighborhood is different from other methods. In the degraded ceiling approach, the algorithm accepts every solution whose objective function is less or equal (for the minimization problems) to the upper limit L, which is monotonically decreased during the search by ΔL. Figure 3.2 presents the chart of the degraded ceiling algorithm.

The initial value of the ceiling (L) is equal to the initial cost function $f(s)$ and only one input parameter ΔL has to be specified. In [1], the authors applied successfully the degraded ceiling on exam timetabling problem. They showed two main properties of the degraded ceiling algorithm :

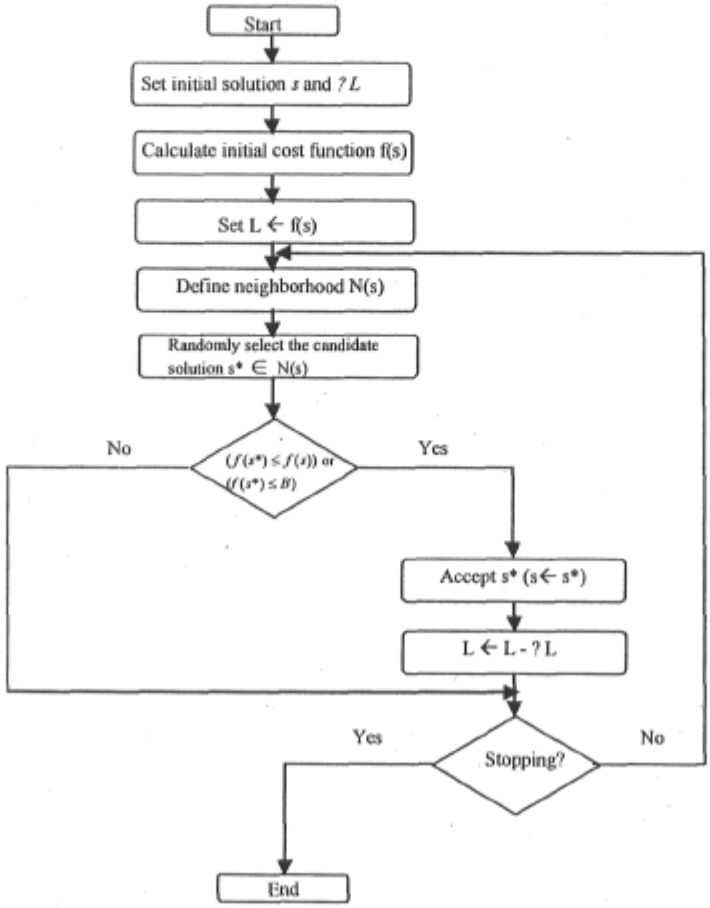

FIG. 3.2 – Degraded ceiling algorithm for minimization problems

- The search follows the degrading of the ceiling. Fluctuations are visible only at the beginning but later ; all intermediate solutions lie close to a linear line.
- When a current solution reaches the value where any further improvement is impossible, the search rapidly converges. The search procedure can then be terminated at this moment.

3.4.3 Degraded ceiling algorithm for buffer allocation problem

In order to adapt the degraded ceiling algorithm to the buffer allocation problem, it is necessary to specify the type of the adopted neighborhood. The search proceeds iteratively from one feasible solution to another by moves in the neighborhood. The move is made as follows. Given an initial solution s with the buffer allocations $B = \{B_1, \ldots, B_{n-1}\}$, two buffer locations p and q ($p, q \in \{1, 2, \ldots, n-1\}$ and $p \neq q$) are

randomly chosen. The corresponding allocations B_p and B_q are modified as follows : $B_p = B_p - 1$ and $B_q = B_q + 1$.

The initial solution is generated by allocating the quantity to all buffer locations and any remaining resources are placed in the middle location. Figure 3.3 shows the pseudocode of the proposed algorithm. Note that in this algorithm, at each step, the ceiling has been increased by ΔL because we have a maximization problem.

1. Set the initial buffers allocation: $B_i \leftarrow K / (n-1)$; $i = 1,\ldots,n-1$ (any remaining resources are placed in the middle location)
2. Initialize ΔL
3. Calculate the objective function $f(s)$
4. Initialize the ceiling $L = f(s)$
5. While the stopping criterion is not reached do

   Randomly choose two buffer locations p and q and set $B_p \leftarrow B_p - 1$ and $B_q \leftarrow B_q + 1$; the new solution is called s*

   if ($f(s) \leq f(s^*)$) or ($f(s^*) \geq L$)

   then accept s*

   Increase the ceiling L ($L = L + \Delta L$)

FIG. 3.3 – Application of degraded ceiling on buffer allocation problem

The approach described above will allow the allocation plan to be calculated subject to a given amount of total buffers slots in a computationally efficient way. The numerical results in section 3.5 will show that within an acceptable amount of time, our algorithm produces better results than simulated annealing. For more details on the simulated annealing algorithm, see [19][20][21].

3.5 Numerical results

In this section, we present the comparison of the performance of the degraded ceiling approach to the simulated annealing algorithm presented in [19][20][21]. All the simulations are made using a 1.8 Ghz Pentium 4 processor. To compare the degraded ceiling and the simulated annealing algorithms, we apply it to exactly the same sets of data (parameters values) and we use the same evaluative algorithm.

To visualize the operations of the two algorithms, we reported in figures 3.4 and 3.5 the values of the cost function (i.e. throughput value) obtained with a production line

of 7 machines. All the machines have the same failure and repair rates (0.3 and 0.57). A total of 30 buffers slots need to be allocated among the production line. Figure 3.4 shows that the search rigidly follows the degraded ceiling, and the point of convergence is quite recognizable (iteration 300). In figure 3.5 (simulated annealing approach), we can see that the oscillation width decreases following the algorithm's exponential cooling schedule and converges towards the optimal value.

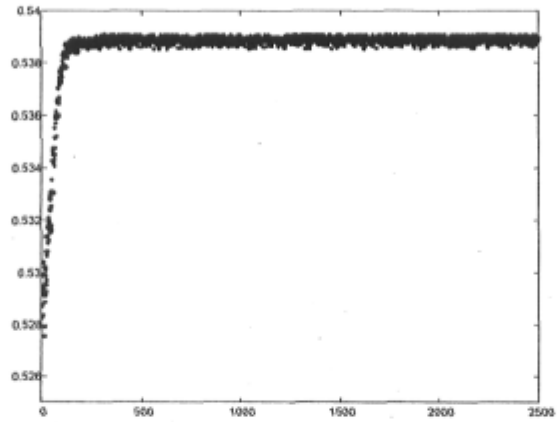

FIG. 3.4 – Solutions with degraded ceiling

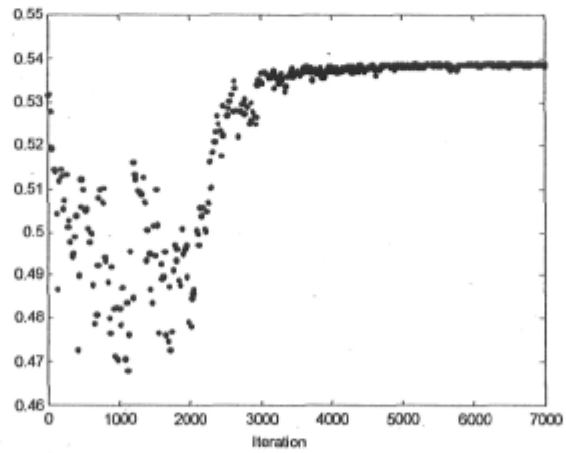

FIG. 3.5 – Solutions with simulated annealing

3.5.1 Example 1

In this example, we consider a non-homogeneous line of 7 machines, initially proposed by [12] and used in [7]. A total of 54 buffers slots need to be allocated among this production line. Table 3.1 shows the data of example 1. In order to compare the results of our approach with those obtained by [12] and [7], we implemented the same evaluative tool (i.e. ADDX algorithm) used in [7].

Machine	**1**	**2**	**3**	**4**	**5**	**6**	**7**
MTTR $= 1/\mu_i$	450	760	460	270	270	650	320
MTTF $= 1/\lambda_i$	820	5700	870	830	970	1900	1100
$\mathbf{T}_i$	40	34	39	38	37	40	43

TAB. 3.1 – Data of example 1 (Ho et al., 1979)

Table 3.2 shows the results obtained by the three methods. The optimal solution is obtained by the degraded ceiling algorithm and by the method proposed by [7]. Note that the execution time to reach the optimal solution is less than one second using our approach. This time depends greatly on the convergence time needed by the ADDX algorithm.

	Location buffer i							
Method	1	2	3	4	5	6	K	**Throughput**
Ho et al.	5	11	8	7	19	4	54	0.0126
Gershwin and Schor	8	10	13	10	9	4	54	0.0128
Degraded ceiling	8	10	13	10	9	4	54	0.0128

TAB. 3.2 – Results of different methods

3.5.2 Example 2

In this example, we consider a homogeneous production line (all the processing time are set to 1) with 10 machines and 9 buffer locations. The number of buffer slots to be allocated among the production line is 270. The evaluative tool used is the DDX algorithm developed in [4]. The search space size is larger than $6,243.10^{19}$. Table 3.3 shows the data of this production line.

By varying ΔL, the number of evaluated solutions needed to converge and the solutions quality change. Figure 3.6 shows the effect of ΔL on the quality of generated solutions.

Machine	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$1/\mu_i$	7	7	5	10	9	14	5	8	10	10
$1/\lambda_i$	20	30	22	22	25	40	23	30	45	20

TAB. 3.3 – Data of example 2

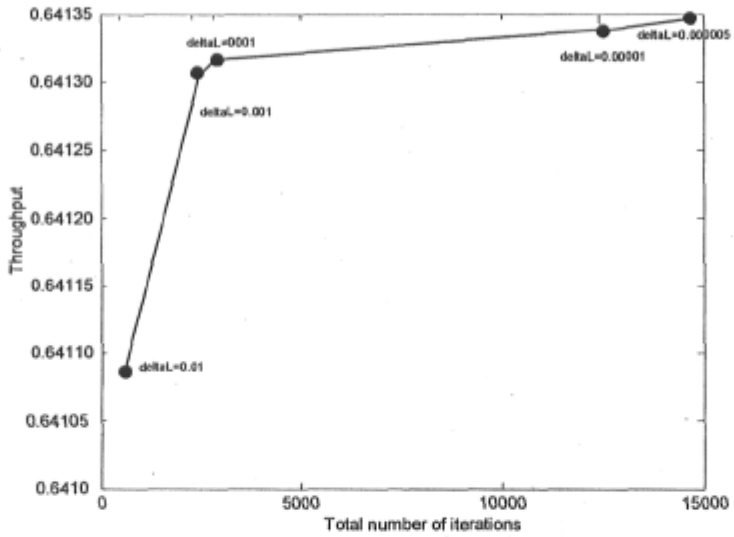

FIG. 3.6 – Influence of ΔL degraded ceiling performance

With $\Delta L = 5.10^{-6}$, we have the best configuration with a throughput equal to 0.64135. The number of iterations needed to reach the optimum configuration does not exceed 15000.

In order to compare the performance of the two approaches (i.e. degraded ceiling and simulated annealing), the simulated annealing algorithm has been implemented with exactly the same parameters values proposed in [19] :

- Maximum trials at given temperature : $100n(= 1000)$.
- Maximum successes at given temperature : $10n$ $(= 100)$.
- Initial temperature : 0.5.
- Cooling schedule : $T_{k+1} = 0.9T_k$.

The parameter ΔL used in the degraded ceiling algorithm has been set to 5.10^{-6}. Figures 3.7 and 3.8 show the convergence curves of the two algorithms. The degraded ceiling algorithm converges, faster (14000 iterations) to the optimal solution than the simulated annealing (45000 iterations). The optimal throughput obtained by the two algo-

rithms is 0.64135 with the following buffer allocation : $B = \{14, 19, 30, 54, 45, 27, 23, 24, 34\}$.

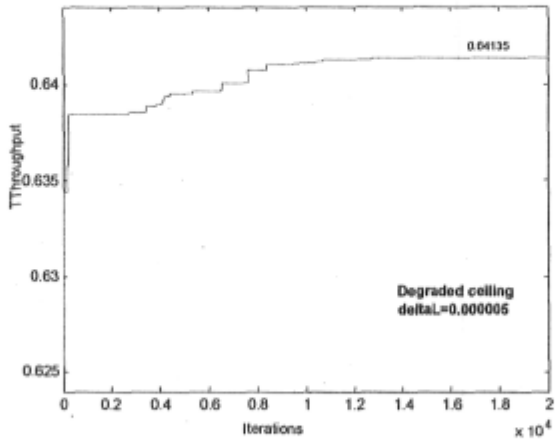

FIG. 3.7 – Evolution of the solution with degraded ceiling

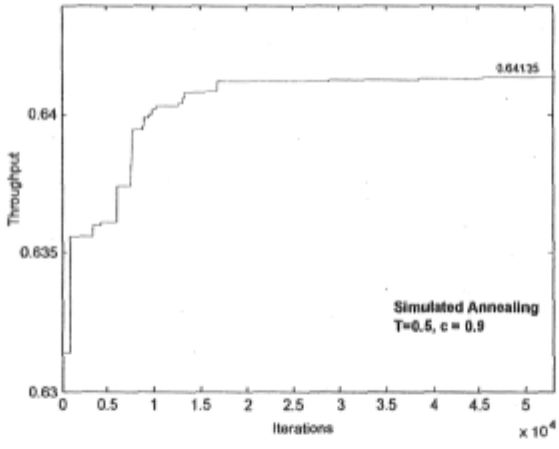

FIG. 3.8 – Evolution of the solution with simulated annealing

The execution time of iteration depends greatly of the DDX convergence time. In this example, the convergence criterion of the DDX algorithm is set to 10^{-4}. In this case, the mean time needed by the DDX algorithm to converge is estimated to 0.08 second. Figure 3.9 shows the effect of the quantity K of buffer slots to allocate among the production line on the number of the evaluated solutions needed to converge.

The quantity K has been varied from 90 to 450. For each value of K, five runs have been made for each approach. As shown in figure 3.9, the number of evaluated configurations needed by the degraded ceiling is lower than those of the simulated

annealing. We can see also that K does not have a significant influence on the number of evaluated configurations for the two algorithms.

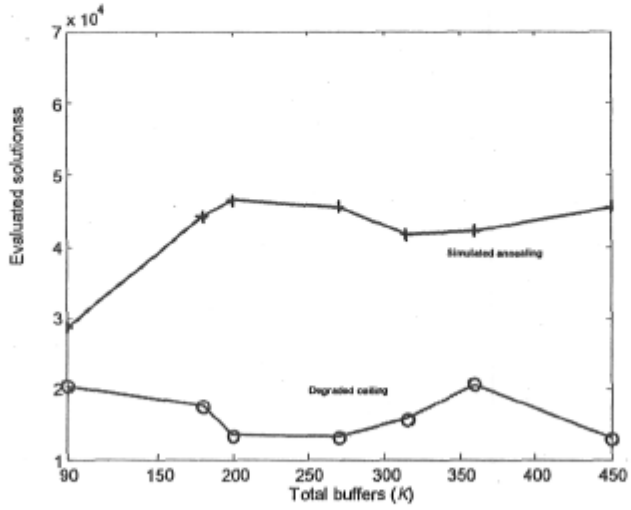

FIG. 3.9 – An overview of an iteration of the hybrid algorithm (ACO/DC).

3.5.3 Long production lines

The two approaches have been tested on homogeneous production lines with different sizes (from 10 to 40 machines). The quantity K to be allocated is set to $(5n)$. Again our objective is to compare the performance of the two approaches (i.e. the quality of the solutions and the number of the evaluated solutions needed by each algorithm). Figure 3.10 shows the number of evaluated solutions needed by the two approaches to converge. It is clear from figure 3.10 that the degraded ceiling algorithm becomes competitive as the size of the production line increases. Both algorithms converge to the same solution in all the experiments. It remains that the advantage of the degraded ceiling approach is that, within an acceptable amount of time, it produces better results than simulated annealing. Furthermore, the best published results are produced more quickly than simulated annealing.

3.6 Conclusion

In this article, we applied a new approach based on the degraded ceiling metaheuristic to solve the buffer allocation problem in the production lines with unreliable

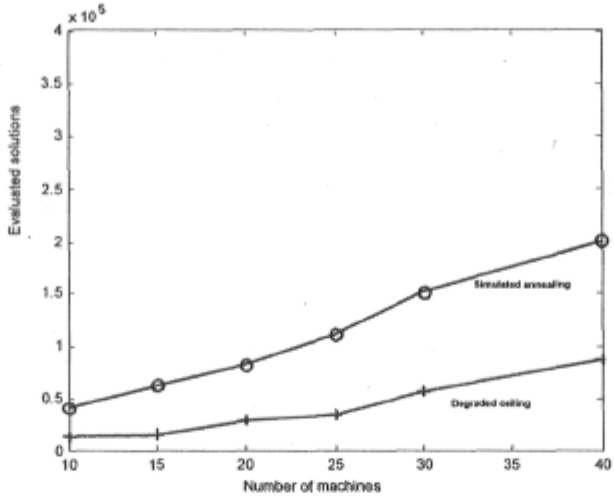

FIG. 3.10 – Number of evaluated solutions needed for different size of production line

machines. In order to prove its efficiency, we compared the degraded ceiling with the simulated annealing algorithm. The results obtained using the degraded ceiling method to solve the problem, are clearly encouraging. The advantage of this method is that it requires the setting of only one parameter that corresponds to a search time. This means that we do not need to launch the algorithm several times to find the best value. In addition, our algorithm shows the trade-off between search time and the quality of the overall allocation plan. Better solutions are produced after a longer search. This property allows the designer to select an acceptable processing time for each particular problem.

The current study makes a comparison between the degraded ceiling and the simulated annealing which is known to be very efficient in solving the buffer allocation problem. While our degraded ceiling approach was shown to produce within an acceptable amount of time better results than simulated annealing, future research should address the degraded ceiling algorithm as compared to other heuristic techniques, such as tabu search, variable neighborhood search, GRASP and beam search, to name a few.

Our future work will also test the method's potential on similar problems especially involving parallel machines. We are currently investigating the issue of hybridization of the degraded ceiling with the ant colony metaheuristic.

Bibliographie

[1] Burke EK, Bykov Y, Newall JP and Petrovic S. *A New Local Search Approach with Execution Time as an Input Parameter.* Computer Science Technical Report No. NOTTCS-TR-2002-3. School of Computer Science and Information Technology. University of Nottingham.

[2] Burman M.H. *New Results in Flow Line Analysis.* Ph. D. Thesis, MIT, Cambridge MA., 1995.

[3] Buzacott J.A. *Automatic transfer lines with buffer stocks.* International Journal of Production Research, 1967, 5(3), 182-200.

[4] Dallery Y, David R, Xie X.L. *Approximate Analysis of Transfer Lines with Unreliable Machines and Finite Buffers.* IEEE Transactions on Automatic Control, 1989, 34, 943-953.

[5] Dallery Y, David R and Xie X.L. *An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers.* IIE transactions, 1988, 20(3), 280-283.

[6] Dallery Y and Le Bihan H. *Homogenisation techniques for the analysis of production lines with unreliable machines having different speeds.* European Journal of Control, 1997, 3, 200-215.

[7] Gershwin S.B and Schor J.E. *Efficient algorithms for buffer space allocation.* Annals of Operations Research, 2000, 93, 117-144.

[8] Gershwin S.B, Schick I.C. *Continuous model of an unreliable two-stage material flow system with a finite interstage buffer.* Tech. Rep. LIDS-R-1039, Massachsetts Institute of Technology, Cambridge MA, 1980.

[9] Gershwin S.B. *An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking.* Operations Research, 1987, 35, 291-305.

[10] Hillier F.S, So K.C. *The effect of the coefficient of variation of operation times on the allocation of storage space in production line system.* IIE Transactions, 1991, 23, 198–206.

[11] Hillier F.S, So K.C, Boling R.W. *Notes : Toward characterizing the optimal allocation of storage space in production line systems with variable processing times.* Management Science, 1993, 39, 126–133.

[12] Ho Y.C, Eyler M.A, Chien T.T. *A Gradient Technique for General Buffer Storage Design in a Production Line.* International Journal of Production Research, 1979, 17, 557-580.

[13] Kirckpatrick S, Gerlatt C.D Jr and Vecchi M.P. *Optimization by simulated annealing.* Science, 1983, 220, 671-680.

[14] Koulmas C., Antony S.R and John R. *A survey of simulated annealing applications to operations research problems.* Omega International Journal of Management Science, 1994, 22, 41-56.

[15] Liu X-G and Buzacott J.A. *Approximate models of assembly systems with finite banks.* European Journal of Operational Research, 1990, 45, 145-154.

[16] Papadopoulos H.T, Heavey C, Browne J. *Queueing Theory in Manufacturing Systems Analysis and Design.* Chapman and Hall, London, 1993.

[17] Pincus M. *A Monte Carlo Method for the Approximate Solution of certain Types of Constrained Optimization Problems.* Oper. Rese., 1970, 18, 1225-1228.

[18] Singh A, MacGregor Smith J. *Buffer allocation for an integer nonlinear network design problem.* Computers & Operations Research, 1997, 24, 453-472.

[19] Spinellis D, Papadopoulos C.T. *A simulated annealing approach for buffer allocation in reliable production lines.* Annals of Operations Research, 2000, 93, 373–384.

[20] Spinellis D, Papadopoulos C.T. *Stochastic Algorithms for Buffer allocation in Reliable Production Lines.* Mathematical Problems in Engineering, 2000, 5, 441-458.

[21] Spinellis D, Papadopoulos C, Macgregor Smith J. *Large production line optimization using simulated annealing.* International Journal of Production Research, 2000, 38, 509- 541.

[22] So K.C. *Optimal Buffer Allocation Strategy for Minimizing Work-in-Process Inventory in Unpaced Production Lines.* IIE Transactions, 1997, 29, 81-88.

Chapitre 4

Selecting machines and buffers in unreliable series-parallel production lines

Résumé

Ce chapitre formule un nouveau problème de conception optimale des lignes de production série-parallèle où des machines redondantes et des stocks tampons sont introduits pour atteindre un meilleur taux de production. L'objectif est de maximiser le taux de production de la ligne sous une contrainte de budget. Les machines et les stocks tampons sont choisis parmi une liste de versions disponibles dans le marché. Les stocks tampons sont caractérisés par leurs coûts et leurs capacités. Les machines sont caractérisées par leurs coûts, leurs taux de production, leurs taux de réparation, leurs taux de panne et leurs temps de traitement. L'approche proposée se distingue des approches existantes par son pouvoir de sélection à la fois des technologies des machines et des capacités des stocks intermédiaires. Pour estimer la performance de la ligne de production série-parallèle, une approximation analytique de type décomposition est proposée. Cette méthode est validée par des résultats de simulation qui montrent qu'elle est très précise. Deux algorithmes utilisant des métaheuristiques, l'un à base des colonies de fourmis et l'autre à base du recuit simulé, sont développés pour la résolution du problème d'optimisation combinatoire résultant. Les résultats numériques obtenus, pour différentes instances générées aléatoirement, illustrent l'efficacité de l'algorithme à colonies de fourmis lorsqu'il est couplé avec une procédure d'amélioration.

4.1 Abstract

This article formulates a new optimal design problem of a series-parallel manufacturing production line, where parallel machines and in-process buffers are included to achieve a greater production rate. The objective is to maximize production rate subject to a total cost constraint. Machines and buffers are chosen from a list of products available in the market. The buffers are characterized by their cost and size. The machines are characterized by their cost, failure rate, repair rate and processing time. To estimate series-parallel production line performance, an analytical decomposition-type approximation is proposed. Simulation results show that this approximate technique is very accurate. The optimal design problem is formulated as a combinatorial optimization one where the decision variables are buffers and types of machines, as well as the number of parallel machines. To solve this problem, ant colony optimization and simulated annealing are compared empirically through several test problems.

4.2 Introduction

Production systems are often organized in manufacturing industry with machines or work centres connected in series and separated by buffers [10][19]. This arrangement is called in the literature a production line, or transfer line, or flow line. In this paper, we mainly use the term manufacturing production line or simply production line. To achieve a greater production rate or to achieve a greater reliability, systems are built with machines in parallel [10][2]. This paper deals with the optimal design of series-parallel production lines which may consist of buffers and of machines with different performances. We consider the practical problem where several versions are available, for each machine, on the market. Its own nominal production rate, reliability and price characterize each version offered by the supplier. Buffers are also chosen from a list of possible buffers which differ by their capacity and cost. In order to find the optimal design structure, the appropriate versions should be chosen from the list of available versions for each *buffer and type of machine*, as well as the *number of parallel machines*. The objective is to maximize the performance of the production line given restrictions on the total system cost.

There is a substantial literature on the analysis of production lines with buffers [10] This literature is mainly concerned with the prediction of performance. Much of it is aimed at evaluating the average production rate (throughput) of a system with specified machines and buffers. In [19], the authors present a set of algorithms that select the minimal buffer space in a flow line to achieve a specified production rate. The algorithms are based on analytical approximations of the Buzacott model of a production line [3][4]. For a review of the literature on production line optimization, the reader is referred to [19]. More recent papers dealing with series and series-parallel production lines include [21][34][11][32][35][1][29][27].

In [27], the authors propose a new local search algorithm, based on the extended great deluge, for solving the classical buffer allocation problem in unreliable series production lines. This problem consists of determining near optimal buffer allocation plans for distributing a certain amount of buffer space among the intermediate buffers of a production line. The goal of [27] is then to choose buffers sizes for a production line by assuming that the machines are specified : the only parameters to find are buffers sizes. In [29], the authors formulate an optimal design problem of a series system with buffers, and develop a heuristic approach based on ant system algorithm to solve it. This problem consists of selecting the best combination of machines and buffers to maximize the system throughput under budget constraint. Only one machine is adopted for each sub-system, and the system is homogeneous, i.e. the processing times are equal

at all machines. A simple analytical decomposition-type approximation is then used to estimate the production line throughput.

The goal of the majority of the existing works is to choose buffers sizes for a production line. Except [29], they all assume that the number of machines is specified, and the only parameters to find are buffers sizes. The present proposed approach to optimal design aims at selecting buffer, machines and the number of parallel machines. It extends [29] in order to take into account parallel machines. That is, while in [29] the authors deal with series lines, the optimal design problem studied in the present work considers series-parallel lines (i.e. each sub-system consists of a set of machines in parallel). This difference in the system structure leads to differences in the optimization method, and in the method used to evaluate the system throughput as well. In [29], the authors developed an algorithm based on ant system which deals only with series lines and cannot be used for series-parallel lines.

To deal with the optimal design problem considered in this work, it is mandatory to develop a new method for throughput evaluation of series-parallel manufacturing production lines. This method has to take into account two characteristics :

i Components may consist of banks of parallel machines. Concerning this first characteristic, we attempt to represent each stage by an equivalent single component.

ii The processing rate may differ from component to component. To deal with this second characteristic, we use a continuous (or fluid) material model type which produces very good results as shown in [9]. This consists of two main steps. First, the non homogeneous line is transformed into an approximately equivalent homogeneous one. In a second step, the resulting homogeneous line is analysed by using the well-known decomposition method for homogeneous lines [15].

The effect of the used simplifications for estimating throughput is examined by comparing the results provided by our approximate technique to simulation results on many examples. This comparison shows that the proposed approximate technique is very accurate.

The problem formulated in this paper is a complicated combinatorial optimization one. It is similar but more complicated than the reliability redundancy allocation problem in series-parallel systems, since it includes also the selection of buffers. The redundancy allocation problem is known to be NP-hard [5]. When selecting machines (versions and redundancy levels) and buffers, the size of the search space is very large even though the problem size is small or moderate. For example, let us consider a production line with 20 components ($n = 20$) in series. Assume that the maximum

number of machines allowed to be connected in parallel is $R_i = 4$, and that the numbers of available versions of machines and buffers are, respectively, $H_i = 5$ and $F_j = 4$ $(i = 1, .., n; j = 1, .., n-1)$. The search space size is given by $s = \left(\prod_{i=1}^{n-1} F_i\right)\left(\prod_{i=1}^{n} H_i.R_i\right)$ and is larger than 2.88 10^{37}.

An exhaustive examination of all possible solutions is not realistic, considering reasonable time limitations. Because of this, we develop two heuristics to solve the problem. The first heuristic is inspired from the ant colony system (ACS) meta-heuristic : a recent kind of biologically inspired algorithms [12][13]. The second proposed heuristic is based on the simulated annealing (AS) meta-heuristic [22].

The remainder of this paper is organized as follows. In section 4.3, the optimal design problem is formulated as a combinatorial optimization one. In section 4.4, the method for evaluating the throughput of series-parallel production lines is presented and validated through simulation results. The ant colony optimization and an improvement procedure are presented in section 4.5 and 4.6, respectively. In section 4.7, the simulated annealing algorithm is presented. In section 4.8, the three algorithms, i.e., ant colony system, ant colony system coupled with the improvement procedure and the simulated annealing, are compared empirically through several test problems. Conclusions are in section 4.9.

4.3 Optimal design problem

Consider the series-parallel production line represented in figure 4.1. The squares represent machines and the circles represent buffers.

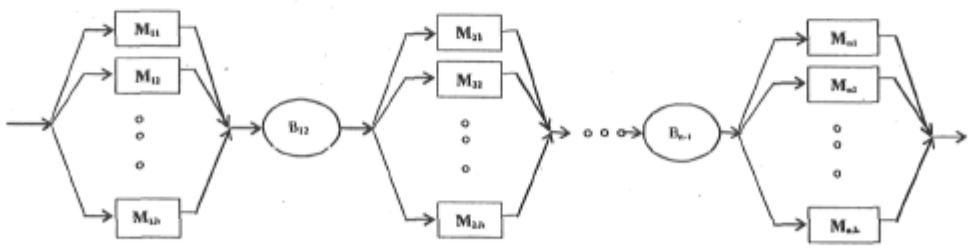

FIG. 4.1 – Series-parallel production line

Buffers are inserted to limit the propagation of disruptions, and this increases the average production rate of the line. This line consists of n components and $n-1$ buffers. Each component of type i $(i = 1, 2, \ldots, n)$ can contain several identical machines

connected in parallel. For each component i, there are a number of machine versions available in the market.

In order to formulate the problem in mathematical expression, the following notations are introduced first :

h_i	version number of machine i
H_i	maximum h_i available
$\mathbf{h}$	$\{h_i\}, h_i \in \{1, 2, .., H_i\}$
r_i	number of elements connected in parallel in i
R_i	maximum r_i allowed
$\mathbf{r}$	$\{r_i\}, r_i \in \{1, 2, .., R_i\}$
$C(h_i)$	cost of each machine of version h_i
$P(h_i)$	isolated production rate of machine with version h_i
$T(h_i)$	processing time of machine with version h_i
$\lambda(h_i)$	failure rate of each machine with version h_i
$\mu(h_i)$	repair rate of each machine with version h_i

We assume that a buffer is also chosen from a list of available buffers. Each version f_i of the buffer i is characterized by a size $N(f_i)$ and a cost $C(f_i)$. The total number of different buffer versions available for the ith component is denoted by F_i. The vector $\mathbf{f} = \{f_i\}$, where $0 \leq f_i \leq F_i$, defines versions of buffers chosen for each component. The entire production line structure is defined by the vectors $\mathbf{h}, \mathbf{r}$ and $\mathbf{f} = \{f_i\}$

For the given $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{f}$, the total cost of the production line can be calculated as :

$$C_T = \sum_{i=1}^{n-1} C(f_i) + \sum_{i=1}^{n} r_i.C(h_i) \tag{4.1}$$

The optimal design problem of production system can be formulated as follows : find the system configuration $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{f}$ that maximizes the total production rate P_T such that the total cost C_T is less or equal to a specified value C^*. That is,

$$\text{Maximize} \qquad P_T(\mathbf{f}, \mathbf{h}, \mathbf{r}) \tag{4.2}$$

$$\text{Subject to} \qquad C_T(\mathbf{f}, \mathbf{h}, \mathbf{r}) \leq C^* \tag{4.3}$$

The input of this problem is C^* and the outputs are the optimal production rate

P_{TMax} and the corresponding configuration determined by the vectors $\mathbf{f}, \mathbf{h}, \mathbf{r}$. The resulting maximum value of P_T is written $P_{TMax}(C^*)$. To deal with this combinatorial optimization problem, a method is presented in the next section to estimate the value of $P_T(\mathbf{f}, \mathbf{h}, \mathbf{r})$.

4.4 Throughput evaluation of series-parallel production lines

4.4.1 Summary of the method

The proposed method approximates each component (i.e. each set of parallel machines) of the original production line as a single unreliable machine. The system is then reduced to a single machine per component production line of the type represented in figure 4.2. The equivalent machines in figure 4.2 may have different processing rates. To determine the steady state behaviour of this *non-homogeneous* production line, it is first transformed into an approximately equivalent homogeneous line. Then, the well-known Dallery-David-Xie algorithm (DDX) is used to solve the decomposition equations of the resulting (homogenous) line [8].

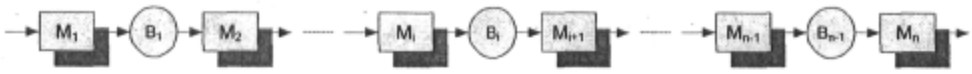

FIG. 4.2 – Equivalent production line

4.4.2 Replacing each component by an equivalent machine

The decomposition techniques developed in the literature are efficient in estimating performance characteristics of series production lines. In these techniques it is necessary for each component to be described by three parameters : *failure rate*, *repair rate* and *processing rate*. By limiting the description of the equivalent machine to these three parameters, our analysis of the new system is reduced in complexity to that of the existing decomposition techniques. Furthermore, the state space of a series-parallel line grows large with the number of parallel machines in the components. Replacing each set of parallel machines by one equivalent machine leads advantageously to a reduction of the state space.

Let denote by λ_{ij}, μ_{ij} and P_{ij}, respectively, the failure rate, the repair rate and the processing rate of a machine M_{ij}, and by λ_i, μ_i and P_i, respectively, the failure rate, the repair rate and the processing rate of a machine M_i. To calculate the three unknown quantities λ_i, μ_i and P_i, we have to formulate three equations. Assuming that machines within the set of parallel machines are fairly similar, the following approximation is proposed in [2] :

$$\lambda_i = \sum_{j=1}^{J_i} \lambda_{ij} \qquad i = 1, 2, \ldots, n \tag{4.4}$$

$$\mu_i = \sum_{j=1}^{J_i} \mu_{ij} \qquad i = 1, 2, \ldots, n \tag{4.5}$$

$$P_i = \sum_{j=1}^{J_i} P_{ij} \qquad i = 1, 2, \ldots, n \tag{4.6}$$

It is shown in [2] that it is a good approximation. However, when buffers are small, this heuristic is inaccurate. In the present work, we assume that the available buffers are large enough. Thus, each set of parallel machines is approximated as an equivalent single machine by using equations (4.4), (4.5) and (4.6). This leads to a non-homogenous line. Therefore, an homogenisation technique is required, as explained in the next subsection.

4.4.3 Homogenisation technique

It is known that in the case of non-homogenous lines (i.e. production lines in which the machines do not have the same processing time), two approaches can be used. The first approach is based on an extension to the case of homogenous lines of the decomposition technique developed in [15] and [6] The second approach relies on the modification of the non-homogeneous line into an approximately equivalent homogeneous line by means of homogenisation techniques [24]. The analysis of the obtained homogeneous line is therefore based on the use of the decomposition method for homogeneous lines. In this way, it is possible to rely on the DDX algorithm which has been proven to be very fast and reliable. In [24], the authors showed that the homogenization method of [26], referred to as the completion time approximation, provides fairly accurate results. In this method, each machine M_i of the original non-homogeneous line is replaced by an approximately equivalent machine M_i^e, such that its completion time distribution is close to that of the original machine. The processing time of machine M_i^e is set to the

processing time of the fastest machine in the original line : $T^e = min(T_1, T_2, \ldots, T_k)$. Since the processing time of is given (equal to T^e), there are two parameters per machine that must be determined, namely the failure and repair rates. Let λ_i^e and μ_i^e be the failure and repair rates of machine M_i^e. The principle of the method developed in [24] is to determine λ_i^e and μ_i^e in such a way that the distribution of completion times of machine M_i^e has the same first and second moments as those of the distribution of completion times of machine M_i. The values of λ_i^e and μ_i^e are given in [24] by :

$$\lambda_i^e = \left[\frac{T_i}{T^e}\left(1+\frac{\lambda_i}{\mu_i}\right)-1\right]^2 \frac{T^e}{T_i}\frac{\mu_i^2}{\lambda_i} \tag{4.7}$$

$$\mu_i^e = \left[\frac{T_i}{T^e}\left(1+\frac{\lambda_i}{\mu_i}\right)-1\right] \frac{T^e}{T_i}\frac{\mu_i^2}{\lambda_i} \tag{4.8}$$

4.4.4 Decomposition equations and DDX algorithm

As said before, we denote by λ_i^e, μ_i^e and T_i^e respectively, the failure rate, the repair rate and the processing time of the machineM_i^e in the equivalent homogenous line. In [8], the authors developed decomposition equations for homogenous lines and propose an efficient algorithm (DDX) to solve these equations.

Production line decomposition methods typically work as follows. An original line is divided into $k-1$ lines with only two machines as illustrated in figure 4.3. The two virtual machines $M_u(i)$ and $M_d(i)$ of line i represent the aggregate behaviour of the production line up-stream and down-stream of buffer B_i. The method requires the derivation of a set of equations that link the decomposed systems together. Such methods are efficient because systems with two machines can be rapidly analyzed. In general, systems may be represented by discrete or continuous flow models. In both, the processing time is deterministic. The discrete material model has the advantage of better representing the discrete nature of typical factories, but it is limited to systems with equal processing times. The continuous (or fluid) model is better suited in our case because it can be used for systems where the machines have different processing rates. The fluid modelling approach is an approximation which consists in using continuous variables to characterize the flow of parts. To intuitively illustrate the fluid modelling point of view, the dynamics of the sand grains in an hourglass can be invoked [7]. The fluid modelling approximation consists in viewing the sand flow in the hourglass as being assimilated to the fluid in a water clock. This is clearly valid as long as we do not turn upside-down the hourglass (i.e. interrupt the flow) with a frequency being of the order of the time required for a single grain to cross the bottleneck region of the

hourglass. Therefore, the quantity of material in each buffer B_i at any time t is a real number taking its value in the interval $[0, N_i]$.

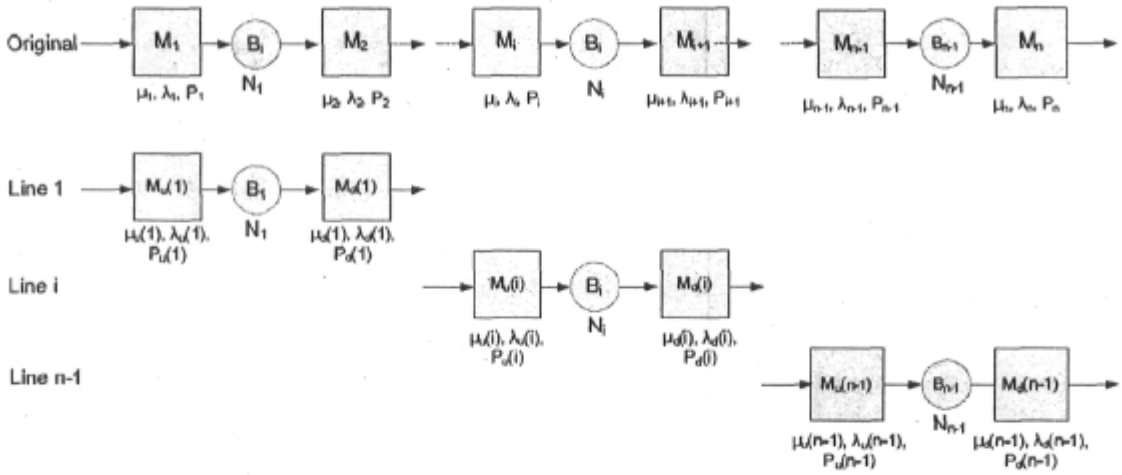

FIG. 4.3 – Decomposition method

The DDX algorithm uses a decomposition method [15] but has greatly reduced the computational complexity. We select the DDX method as our evaluation model because this method can obtain the throughput quite accurately and quickly [31]. In our optimal design problem, the DDX algorithm is used to solve the decomposition equations for each configuration.

To examine the effect of the used simplifications for estimating throughput, the results provided by our approximate technique were compared to simulation results on many examples. This will be illustrated in the next subsection.

4.4.5 Numerical experiments

Simulation models

The simulation models of the series-parallel systems were developed using the commercial package ProModel. The failure and repair rates were all assumed to have an exponential distribution. Each model was run for a 6000 time units warm-up period to reach steady state. We then collected the throughput for the next 60000 time units. We ran each simulation 3 times for each system. The second step of our study was to apply the proposed approximation (i.e. replace each component by an equivalent machine, homogenize the line and apply the decomposition method) on the generated examples and

then compare the obtained results with those obtained by simulating the original systems. The % error in throughput (P_T) is calculated by $Error(\%) = \frac{P_T(A)-P_T(S)}{P_T(S)}.100(\%)$ where $P_T(A)$ and $P_T(S)$ are the throughputs estimated from the analytical method and the simulation respectively. Simulation models were developed for three typical examples taken from [2], and for 200 larger lines which we have randomly generated. The simulation results are presented in the next two subsections.

First experiments

We first consider three typical examples of series-parallel production lines taken from [2]. Each line has three components and two buffers. As shown in figure 4.4, component 2 contains two parallel machines and components 1 and 3 have one machine each. The parameters for each example are given in table 1. For each example, $i = 0.1$ and buffer sizes are set to 10. Examples 1, 2 and 3 correspond, respectively, to typical situations of identical, slow and unreliable machines.

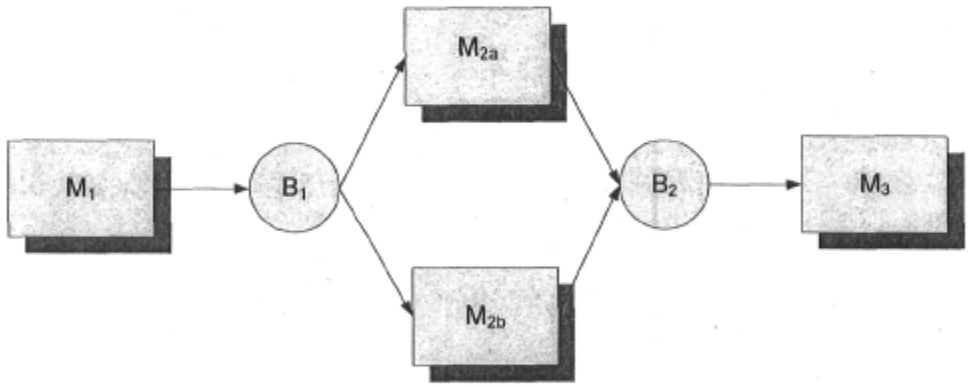

FIG. 4.4 – Typical example of series-parallel production line

The parameters for each example are given in table 4.1. The first example (redundant machines) highlights a line in which both machines of component 2 are identical in all respects to the first and third machines. In the second example (slow machines), both machines of component 2 are identical in all respects to the first and third machines except that each machine in component 2 operates at half the speed of the first and third machines. The third example (unreliable machines) highlights a line in which both machines of component 2 are identical in all respects to the first and third machines except that the failure rate has been increased so that each machine in component 2 is less reliable than the first and third machines.

	Example 1		Example 2		Example 3	
i	λ_i	P_i	λ_i	P_i	λ_i	P_i
1	0.01	1	0.01	1	0.01	1
2a	0.01	1	0.01	0.5	0.12	1
2b	0.01	1	0.01	0.5	0.12	1
3	0.01	1	0.01	1	0.01	1

TAB. 4.1 – Parameters data for three typical examples taken from (Burman, 1995)

	Simulation results	Proposed approximation	Error (%)
Example 1	0.87	0.878	0.91
Example 2	0.831	0.83	-0.12
Example 3	0.756	0.722	-4.5

TAB. 4.2 – Comparison of the analytical method and simulation for three typical examples

Table 4.2 shows the comparison of the throughput from our analytical method and the simulation. The average absolute value of the % error in the throughput estimation is less than 1% for examples 1 and 2 and it is 4.5% for example 3.

Random examples

To verify the efficiency of the suggested approximation method on larger series-parallel lines, we have randomly generated four sets of 50 examples (a total of 200 lines). Each set corresponds to a different length of series-parallel systems (i.e., $n = 4, 10, 15$ and 20). Each example is characterized by :

- A randomly generated number of machines in parallel at each component (between 1 and 4).
- A randomly generated capacity of intermediate buffers (between 30 and 80).
- To obtain random, yet realistic lines, we followed the procedure proposed in [2] to generate the parameters data of machines.

Figures 4.5-4.8 show the % errors in throughput for $n = 4, 10, 15$ and 20. From these results, we can extract the conclusions presented in Table 4.3.

From all the above numerical and simulation experiments, we conclude that the proposed analytical approximation provides good results, i.e., which are accurate enough.

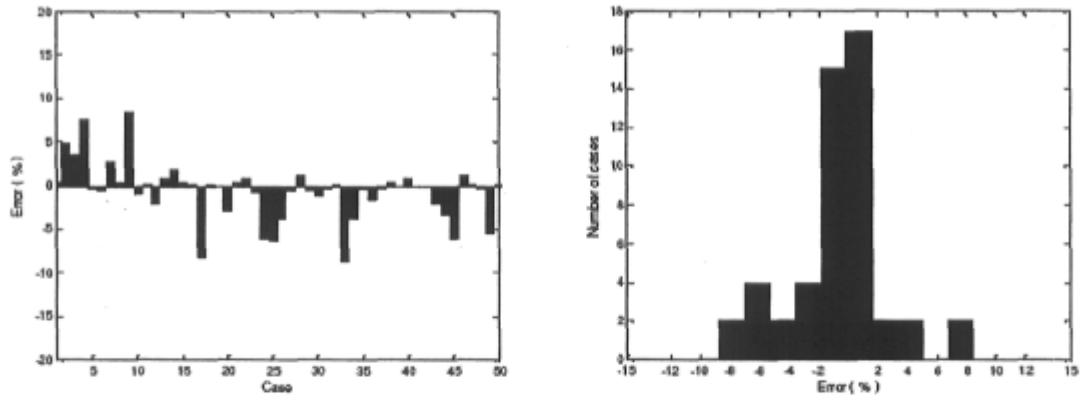

FIG. 4.5 – Throughput error for $n = 4$

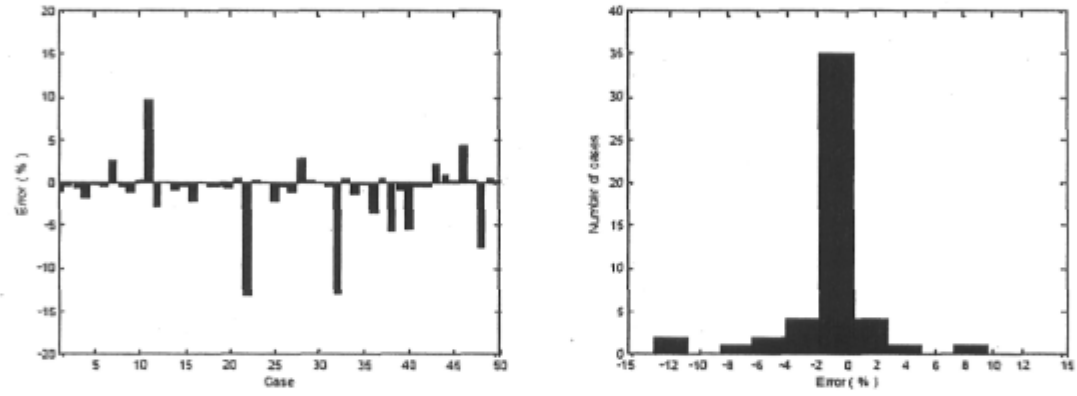

FIG. 4.6 – Throughput error for $n = 10$

Therefore, the method can be safely used in solving the optimal design problem formulated in section 4.3. This method is used in the next section to evaluate the objective function when solving the design problem by ant colony optimization and simulated annealing.

4.5 ACS meta-heuristic for the optimal design of series-parallel lines

Many researchers have shown that insect colonies behaviour can be seen as a natural model of collective problem solving. The analogy between the way ants look for food and combinatorial optimization problems has given rise to the ACS computational

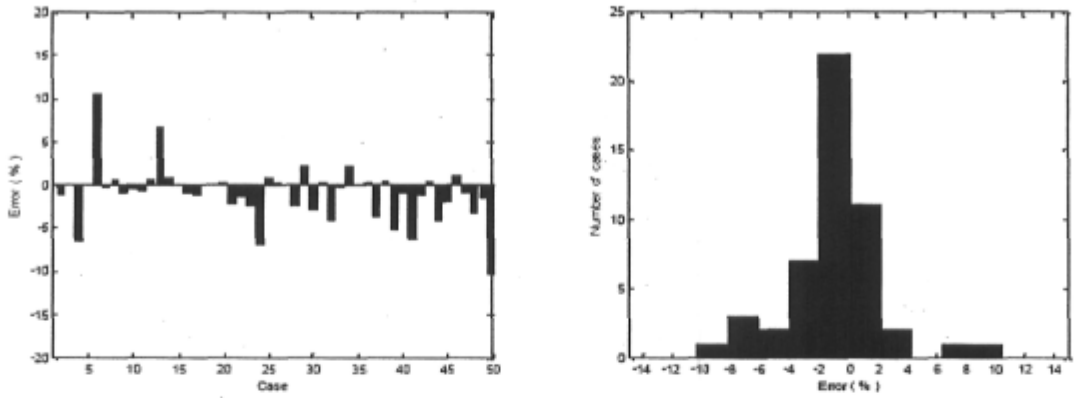

FIG. 4.7 – Throughput error for $n = 15$

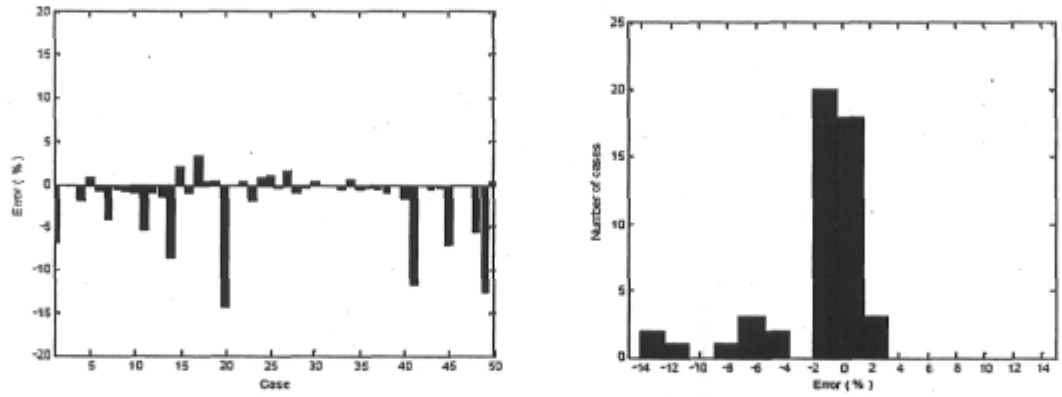

FIG. 4.8 – Throughput error for $n = 20$

paradigm.

4.5.1 Applying ACS to select machines and buffers : the general algorithm

Following [12], with respect to the problem of selecting machines and buffers in a series-parallel line, each ant is an agent that leaves a pheromone trail, called a trace, on the edges of a graph representing the problem. To represent our problem as such a graph, we introduce the following sets of nodes and edges :

– Three sets of nodes :

n	Average error	% of results with error between	
	%	−2% and 2%	−4% and 4%
	2.04	50	66
	1.86	58	76
	2	50	70
	2.1	80	84

TAB. 4.3 – Throughput errors

1. The first set of nodes (N_1) represents the components and the buffers.
2. The second set (N_2) represents the versions of elements available for each component and buffer.
3. The third set of nodes (N_3) represents, for each component, the numbers of elements which can be used in parallel. For example, if the maximum number of elements in parallel is two, the set N_3 will be given by two nodes corresponding to one element and two parallel elements.

– Two sets of edges :

1. The first set of edges is used to connect each node in the set N_1 to the corresponding nodes in N_2.
2. The second set of edges is used to connect some nodes in N_2 to the nodes in N_3.

Figure 4.9 illustrates such a graph for a simple trivial case of two components and one buffer such that :

– Two versions are available for each component and buffer, and
– The maximum number of elements allowed in parallel is two.

Informally, our algorithm works as follows : m ants are initially positioned on a node representing a component. Each ant represents one possible structure of the entire system. This entire production line structure is defined by the vectors $\mathbf{f}, \mathbf{h}$ and $\mathbf{r}$. Each ant builds a feasible solution (called a tour) to our problem by repeatedly applying three different stochastic greedy rules (i.e., the state transition rules). While constructing its solution, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule. Once all ants have terminated their tour, the amount of pheromone on edges is modified again (by applying the global updating rule). Ants are guided, in building their tours, by both heuristic information (they prefer to choose "less expensive and more efficient edges"), and by pheromone information.

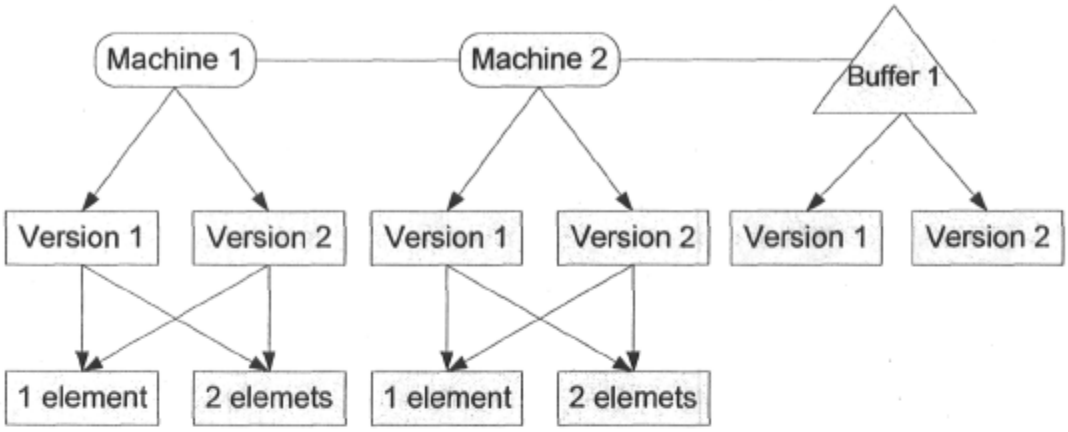

FIG. 4.9 – Graph for a simple trivial case of two components and one buffer

Ants can be guided, in building their tours, by pheromone information and heuristic information. Naturally, an edge with a high amount of pheromone is a very desirable choice. The pheromone updating rules are designed so that they tend to give more pheromone to edges which should be visited by ants.

Note that when an ant builds a solution, it can be feasible or unfeasible. When the obtained solution is unfeasible, it is automatically rejected and it is not taken into account in the comparison with the other feasible solutions obtained by the other ants. It should be noted also that the global update of the pheromone is done only for the best obtained feasible solution. The overview of the algorithm is reported in figure 4.10.

In the following we discuss the state transition rules, the global updating rule, and the local updating rule.

```
Initialize
Loop
   Loop
      Each ant applies a state transition rule to build a solution
      and a local pheromone updating rule is applied
   Until all ants built a complete solution
   Evaluate all solutions and record the best feasible one
   A global pheromone updating rule is applied
Until a stopping criterion is reached
```

FIG. 4.10 – Overview of the algorithm

4.5.2 State transition rules

In the above algorithm, at each step of the construction process, ants use : (1) pheromone trails (denoted by τ_{ij}) to select the versions of machines and buffers and the number of machines connected in parallel ; (2) a problem-specific heuristic information (denoted by μ_{ij}). The value of depends μ_{ij} of the nature of the node (i.e. machine's version or buffer's version). Note that the choice of the number of machines to be connected in parallel is not function of the heuristic information μ_{ij}.

An ant positioned on node i (representing a machine or a buffer) chooses the version j ($j = h_i$ if i is a machine and $j = f_i$ if i is a buffer) according to following :

$$j = \begin{cases} \arg\max_{k \in \Gamma_i} \left\{ (\tau_{ik})^{\alpha} (\eta_{ik})^{\beta} \right\} & \text{if } q \leq q_0 \\ J & \text{Otherwise} \end{cases} \tag{4.9}$$

where Γ_i is the set of nodes representing the available versions for node i ($\Gamma_i = \{1, \ldots, H_i\}$ if i is a machine or $\Gamma_i = \{1, \ldots, F_i\}$ if i is a buffer).

and J is a random variable selected according to the probability distribution given by :

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum_{k \in \Gamma_i} (\tau_{ik})^{\alpha} (\eta_{ik})^{\beta}} & \text{if} \quad j \in \Gamma_i \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

In the above equations (4.9) and (4.10), α and β are parameters that control the relative weight of the pheromone (τ_{ij}) and the local heuristic (η_{ij}), respectively. The value of depends η_{ij} on the type of a given node. The variable q is a random number uniformly distributed in $[0, 1]$; and q_0 is a parameter ($0 \leq q_0 \leq 1$) which determines the relative importance of exploitation versus exploration if i represents a machine and if i represents a buffer.

Similarly, when an ant is positioned on node i representing a version of a machine, it has to select a number j of machines to be connected in parallel. In this case, the used rule is similar to (4.9) and (4.10) except for the heuristic information which is set to 1 and $\Gamma_i = \{1, \ldots, R_i\}$.

4.5.3 Global updating rule

Once all ants have built a complete solution, pheromone trails are updated. Only the globally best ant (i.e., the ant which constructed the best design solution from the beginning of the trial) is allowed to deposit pheromone. A quantity of pheromone $\Delta\tau_{ij}$ is deposited on each edge that the best ant has used, where the indices i and j refer to the edges visited by the best ant. The quantity $\Delta\tau_{ij}$ is given by the total production rate PT_{best} of the design feasible solution constructed by the best ant. Therefore, the global updating rule is :

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{4.11}$$

where $0 < \rho < 1$ is the pheromone decay parameter representing the evaporation of trail.

Global updating is intended to allocate a greater amount of pheromone to greater design solution. Equation (4.11) dictates that only those edges belonging to the globally best solution will receive reinforcement.

4.5.4 Local updating rule

While building a solution of the problem, ants visit edges on the graph G, and change their pheromone level by applying the following local updating rule :

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0 \tag{4.12}$$

where τ_0 is the initial value of trail intensities.

The application of the local updating rule, while edges are visited by ants, has the effect of lowering the pheromone on visited edges. This favours the exploration of edges not yet visited, since the visited edges will be chosen with a lower probability by the other ants in the remaining steps of an iteration of the algorithm.

4.6 Improving constructed solutions

It is well known that the performance of ACS algorithms can be greatly improved when coupled to local search algorithms [14]. Following this idea, an improvement procedure is included in our ACS algorithm once all ants have terminated their tour and before applying the global updating rule.

This procedure consists of two steps :

(1) The remaining budget (the amount not used by the ant) of the obtained structure is first used to improve the solution. In fact, some generated feasible solutions do not use the entire available budget. The procedure (detailed in figure 4.11) improves the initial solution by using this remaining budget to increase the number of machines connected in parallel.

(2) In this step, two types of evaluation are done depending of the nature of the component (i.e. machine or buffer). For each pair of components representing the machines, the number of machines is changed by adding one for the first component and subtracting one for the second component. In the case of buffers, the algorithm proceeds to change in turn each pair of chosen versions by another pair.

The above steps are illustrated in the following example. Let us consider a series-parallel line with 3 machines (5 available versions for each machine) and 2 buffers (6 available versions for each buffer). Suppose that the solution at a given cycle is given by $f = \{3, 2\}$, $h = \{2, 1, 1\}$ and $r = \{2, 3, 3\}$. The improvement procedure will evaluate the structures with the following numbers of parallel machines :

$r = \{1, 4, 3\}$, $r = \{1, 3, 4\}$, $r = \{2, 2, 4\}$, $r = \{3, 2, 3\}$, $r = \{3, 3, 2\}$ and $r = \{2, 4, 2\}$, and the following versions of buffers : $f = \{2, 3\}$ and $f = \{4, 1\}$.

Note finally that when this improvement procedure is used, only the neighbouring feasible solutions are evaluated and compared with the current solution.

The overview of the improvement algorithm is reported in figure 4.11.

```
Step one
1.  Start with the first component (representing the machines)
2.  If the remaining budget is sufficient, increase the number of   machines connected in
    parallel by one.
3.  Update the remaining budget.
4.  Apply DDX algorithm with the homogenization technique and record the obtained
    solution.
5.  If the budget is sufficient, switch to the next element and go to 2. Otherwise stop.

Step two
/*Machines structure*/
1.  Start with the first component representing the machines (i=1)
2.  If i < n, change the number of machines connected in parallel by adding one machine
        (resp. subtracting one).
3.  For each subsequent component j (j=i+1,..,n):
        - Change the number of machines connected in parallel by subtracting one machine
        (resp. adding one).
        - Apply DDX algorithm with the homogenization technique and record the obtained
        solution.
4.  Set i=i+1 and go to 2.

/*Buffers structure*/
5.  Start with the first buffer (i=1)
6.  If i < n-1, change the chosen version f_i by another version f_i+1(resp. f_i-1).
7.  For each subsequent buffer j (j=i+1,..,n-1):
        - change the chosen version f_j by another version f_j-1(resp. f_j+1).
        - Apply DDX algorithm with the homogenization technique and record the obtained
        solution.
8.  Set i=i+1 and go to 6.
9.  Evaluate the obtained solutions and update the initial solution if improved
```

FIG. 4.11 – Improvement procedure

4.7 Simulated annealing algorithm for the optimal design of series-parallel lines

The simulated annealing (SA) exploits the analogy between the way in which a metal cools and freezes into a minimum crystalline energy and the search for a minimum in a more general energy. The connection between this algorithm and mathematical minimization was first noted by Pincus [30] but it was Kirckpatrick et al. [22] who proposed that it form the basis of optimization technique for combinatorial problems. An overview of its different applications is given in [23].

The simulated annealing technique is an optimization method suitable for combinatorial minimization problems. A new solution is generated and compared against the current solution. The new solution is accepted as the current solution if the difference in quality does not exceed a dynamically selected threshold. The solutions corresponding to larger increases in cost have a small probability of being accepted. A parameter that regulates the threshold is called temperature and the function that determines the values for the temperature over time is called the cooling scheduling. The temperature decreases over time to decrease the probability of non improving moves.

The simulated annealing proposed for the design of series-parallel production lines is given in figure 4.12.

1. Set initial solution S_0 and set $S \leftarrow S_0$.
2. Set initial temperature T and cooling rate c ($0 \leq c \leq 1$).
3. Initialize the inner loop step count $U \leftarrow 0$ and success count $V \leftarrow 0$.
4. While ($T > T_{min}$)
   a) Create a new solution S' from the neighborhood space and set $\Delta S \leftarrow S - S'$.
   b) If S' is feasible and $\Delta S < 0$, accept the new solution (i.e. $S \leftarrow S'$) and set $V \leftarrow V+1$.
   c) If S' is feasible and $\Delta S \geq 0$, accept the new solution with the probability $\exp(-\Delta S/T)$ and set $V \leftarrow V+1$ (when accepted).
   d) If $V = V_{max}$ then go to step f).
   e) Repeat for current temperature. Set $U \leftarrow U + 1$. If $U <$ maximum number of steps, go to step a).
   f) Decrease the annealing temperature ($T \leftarrow c.T$) and set $U \leftarrow 0$, $V \leftarrow 0$.
5. End While.

FIG. 4.12 – Simulated annealing algorithm for series-parallel production lines design problem

Initial feasible solution

The initial feasible solution can be generated in many ways. We tried two generation methods. The first one generates a feasible initial solution by taking the least expensive solution (i.e. only one machine in each component and version 1 for all buffers and machines). The second way starts with the less expensive solution and tries to improve it by an iterative improvement procedure. The experimental tests show that the first method is better.

Neighboring solution

There are many ways to define neighborhood for this problem. On the one hand, two types of neighbourhood structures have been tested. Regarding the number of machines in parallel for example, the first type was adding or subtracting one machine. The second one consisted in choosing a random number of machines in parallel. This kind of neighbourhood move has been proposed also in [33] when solving buffer allocation problem. The carried out experiments showed that the second way is slightly more effective. On the other hand, the versions of the machines are indexed in ascending order of the production rate P(hi).

Our adopted neighboring structure can be summarized by the following steps :

Step 1. Randomly select a component *comp* representing either a machine or a buffer.

Step 2. If $comp = machine$, randomly select one of these two actions :

Action 1 : Change the number of machines in parallel by choosing a random number less than k_{max}(k_{max} is the maximum number of machines allowed to be connected in parallel).

Action 2 : Change the version of machine $version(machine)$ by $version(machine)+1$ or $version(machine) - 1$.

Step 3. If $comp = buffer$, change its version $version(buffer)$ by $version(buffer)+1$ or $version(buffer) - 1$.

When a neighbour solution is randomly selected, it can be either feasible or unfeasible. If the solution is unfeasible, it is automatically rejected without using the criterion of acceptance and the algorithm passes to the next iteration.

Before introducing the numerical results, it should be noted that it would be straightforward to iterate the improvement procedure until no further improvements are found, i.e. to turn it into a local hill-climber. The coupling of a local search procedure such as the hill-climbing with the ACES may give a good idea on the quality of the obtained solutions. However, this will increase considerably the total computation time. Because the calculation of the objective function depends greatly on the convergence of the DDX algorithm whose time is not negligible, we proposed a local search procedure which does not require much evaluations of the objective function as the hill-climbing.

4.8 Numerical results

To prove the efficiency of our algorithm when combined with the local search, we proposed four examples of production line with respectively 4, 10, 15 and 20 components. Tables 6-13 show the corresponding data. The versions are indexed in ascending order of the production rate $P(h_i)$. The available budgets are respectively 160\$, 300\$, 450\$ and 750\$ and the maximum number of machines allowed to be connected in parallel is 3 for the first example and 4 for the rest. The search space size is respectively larger than $5.5\ 10^5, 2.684\ 10^{18}, 8.796\ 10^{27}$ and $2.88\ 10^{37}$. All the algorithms were implemented using MATLAB on a PC with 1.8 Ghz processor.

We implemented the simulated annealing and ACS algorithm and ran simulations to set the parameters. For the ACS algorithm, the parameters considered are those that effect directly or indirectly the computation of the probability in formulas (4.9) and (4.10) (i.e. $\alpha, \beta, \rho, \tau_0$ and q_0). Following the tuning procedure used in [12][13][28], we tested several values for each parameter, while all the others were held constant (over ten simulations for each setting in order to achieve some statistical information about the average evolution). Based on these initial experiments the values found to be most appropriate are given in Table 4.4. Furthermore, in all our experiments, the number of ants is set to 5. Note that the ACS is not very sensitive to changes in these values, and tested well for quite a range of them. The parameters considered for the simulated annealing are the initial temperature T, length of the inner loop, the final temperature T_{min}, the maximum of success solution V_{max} and the cooling rate c. Initially, the temperature T is set to a sufficiently high value to accept all solutions during the initial phase of the simulated annealing. The cooling rate c should be generally greater than 0.7. Table 4.4 shows the values of all the parameters considered in the three algorithms.

Each algorithm was tested by performing ten trials. Figures 4.13-4.16 show the highest throughput versus the number of evaluations. By 6000, 200.000, 250.000 and 400.000 evaluations of throughput respectively for example $1, 2, 3$ and 4, the highest throughput has leveled out. These numbers of evaluations are used to assess the performance of the algorithms.

[0](*) $\beta = \beta_1$ when the node i is a machine and $\beta = \beta_2$ otherwise.

	Example 1 ($n = 4$)	Example 2 ($n = 10$)	Example 3 ($n = 15$)	Example 4 ($n = 20$)
ACO*	$\alpha = 0.1, \beta_1$=0.5 β_2= 1, ρ =0.01 τ_0=$\frac{1}{25}$; q_0=0.75	$\alpha = 0.1, \beta_1$=0.1 β_2= 0.3, ρ =0.01 τ_0=$\frac{1}{25}$, q_0=0.75	$\alpha = 1, \beta_1$=1.5 β_2= 1, ρ =0.05 τ_0=$\frac{1}{15}$, q_0=0.80	$\alpha = 1, \beta_1$=1.5 β_2= 1, ρ =0.05 τ_0=$\frac{1}{15}$, q_0=0.80
ACO with improving procedure*	α =0.1, β_1=0.1 β_2=1, ρ =0.05 τ_0=$\frac{1}{25}$, q_0=0.80	α =0.5, β_1=1 β_2=1, ρ =0.05 τ_0=$\frac{1}{50}$, q_0=0.80	α =0.5, β_1=1 β_2=1, ρ =0.05 τ_0=$\frac{1}{50}$, $q0$ =0.75	α =1, β_1=0.3 β_2=0.5, ρ =0.05 τ_0=$\frac{1}{35}$, $q0$ =0.75
Simulated annealing	c =0.98, T =5 inner loop=40 T_{min}=5.10^{-5} V_{max}=40	c =0.99, T =5 inner loop=100$*n$ T_{min}=5.10^{-5} V_{max}=35$*n$	c =0.95, T =5 inner loop=100$*n$ T_{min}=5.10^{-5} V_{max}=35$*n$	c =0.975, T =5 inner loop=100$*n$ T_{min}=5.10^{-5} V_{max}=35$*n$

TAB. 4.4 – Parameters values

	$P_{T_{Max}}$	C_T(\$)	H	r	f
Example 1	4.7074	160	{1311}	{3333}	{212}
Example 2	3.2467	250	{1111551212}	{2223112111}	{111131111}
Example 3	4.4406	450	{111111111111211}	{322422222222422}	{11211111111211}
Example 4	3.8991	750	{24315424121212112342}	{32242222423232432222}	{12211312211111112111}

TAB. 4.5 – Configurations results for examples 1,2,3 and 4

	Simulated annealing					ACS					ACS with improving procedure				
	Min	Mean	STD	Max	tmean	Min	Mean	STD	Max	tmean	Min	Mean	STD	Max	tmean
Example 1	4.7074	4.7074	0	4.7074	<1	4.7074	4.7074	0	4.7074	<1	4.7074	4.7074	0	4.7074	<1
Example 2	3.1162	3.1694	0.0478	3.2467	352 s	3.1162	3.1291	0.0522	3.2282	1130 s	3.2452*	3.2463*	0.0005*	3.2467*	186*s
Example 3	3.5855	3.7107	0.1243	3.8282	933 s	3.5855	3.6649	0.1411	3.9738	1644 s	3.8282*	4.0746*	0.1274*	4.4406*	788*s
Example 4	2.9096	3.0663	0.1252	3.2206	1172 s	2.4375	2.9607	0.3082	3.2325	3960 s	3.2169*	3.4577*	0.2154*	3.8991*	2030*

TAB. 4.6 – Results for examples 1,2,3 and 4

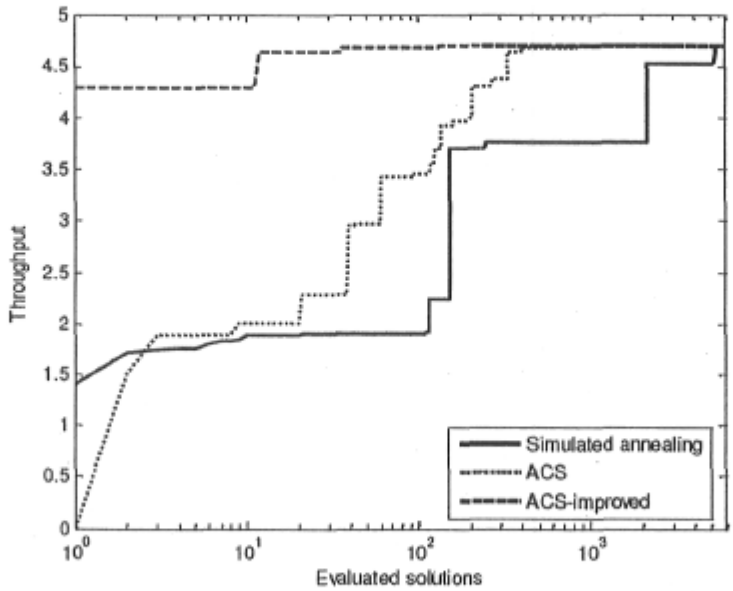

FIG. 4.13 – Convergence results for example 1

The convergence curves in figures 4.13-4.16 show that the ACS algorithm performs better when coupled with the improvement procedure. Generally, the convergence is faster and the quality is better than the other algorithms. The ACO algorithm when coupled with the local improvement procedure starts with a good solution, because the initial solutions built by the ants are improved by the procedure at the first iteration and before being reported in the graph. It is important to note that all the evaluated solutions are taken into account including those generated by the local improvement procedure. The results obtained by simulated annealing and ACS without the improvement procedure are fairly similar in the 4 examples.

The results obtained after 10 trials are given in tables 4.5 and 4.6. The solutions obtained by the ACS when coupled with the improvement procedure are the best obtained solutions. The application of the improvement procedure with the ACS improves the quality of solutions and the time required to produce near optimal solutions. Table 4.6 shows that the best results obtained by the simulated annealing and ACS (without the improvement procedure) are almost similar. However, we remark that :

i. The mean values of the results obtained by the simulated annealing are clearly better than those obtained by the ACS algorithm.
ii. The execution times of simulated annealing and ACS when coupled with the improvement procedure are lower that the execution time of ACS alone. For instance, in example 4 the mean execution time is 3960 seconds for ACS alone and

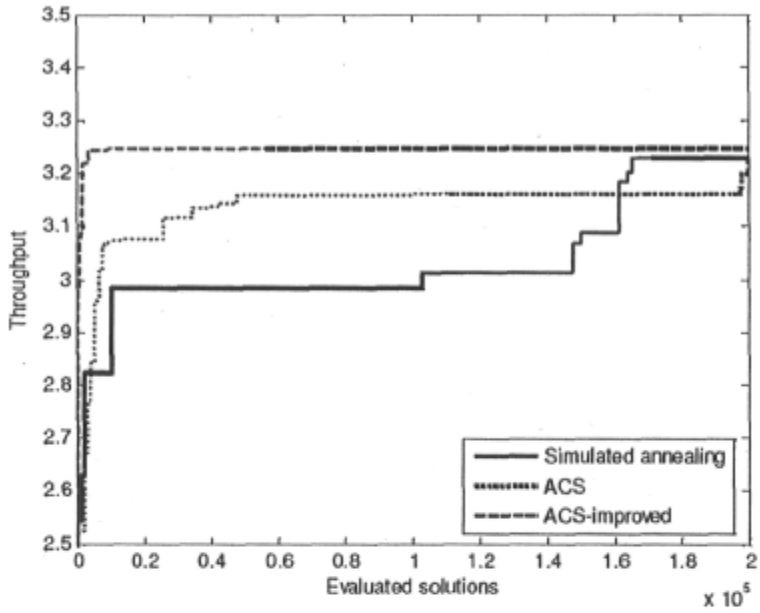

FIG. 4.14 – Convergence results for example 2

it is about 1172 and 2030 seconds for SA and ACS coupled with the improvement procedure, respectively.

In order to compare the performance of the three algorithms, the stopping criterion is the number of evaluated solutions. The computation time of the ACS algorithm, for the same number of evaluated solutions, is higher than that of the other algorithms. This is because the ACS algorithm constructs an *entire solution* (i.e., selects versions and number of machines for each sub-system), at each iteration and for each ant. It implies that a complete loop is used. Thus, each solution construction requires considerable computation time. On the other hand, when the ACS is coupled with the improvement procedure, each generated solution by an ant can be improved by evaluating the neighbour solutions while carrying out *minor changes in the current solution*. Consequently, since it does *not require a complete construction of the solution*, the computation time is decreased. On the other hand, the simulated annealing algorithm constructs the solutions by making minor changes in the current solutions, requiring less computation time than the ACS algorithm when coupled or not with the improvement procedure.

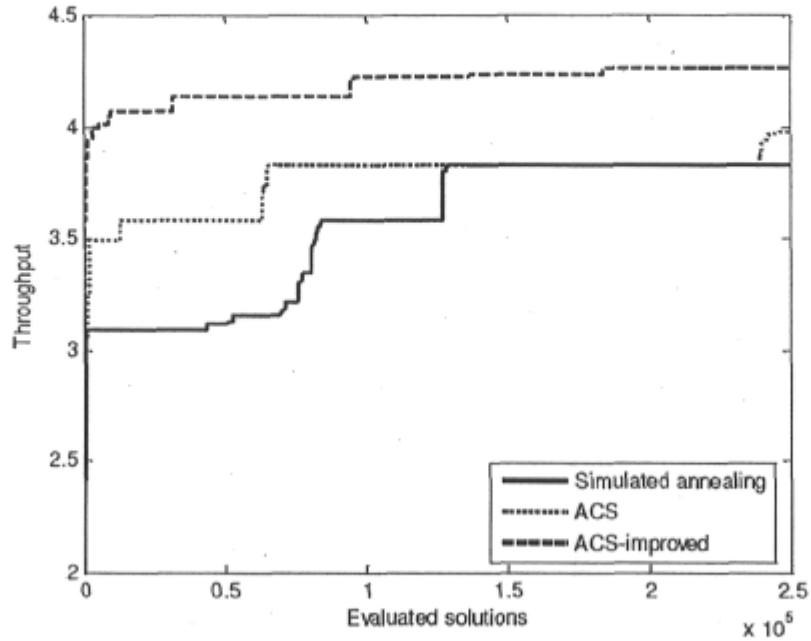

FIG. 4.15 – Convergence results for example 3

Additional tests

A set of 10 test instances are also randomly generated for $n = 20$ and used to evaluate the performance of the proposed algorithms. Note that the parameters used for these 10 test instances are those set by using example 4 as a typical problem (see Table 3, for $n = 20$). By running the algorithms without further tuning on the 10 test instances, we avoid any parameters over-fitting.

The proposed algorithms are evaluated in terms of solutions quality. For each instance, five trials are performed. It has been observed again for these randomly generated instances that the ACS coupled with the improvement procedure (ACS-I) out-performs ACS and SA algorithms. Therefore, let us consider ACS-I as a reference level and let define average errors (relatively to this reference level) as follows :

$$Average\ error(\%) = \frac{Mean(throughput\ SA \text{ or } ACS) - Mean(throughput\ ACS - I)}{Mean(throughput\ ACS - I)}$$

where $Mean(throughput ACS - I)$ is the mean throughput value obtained by the ACS when coupled with the improvement procedure and $Mean(throughput SA \text{ or } ACS)$

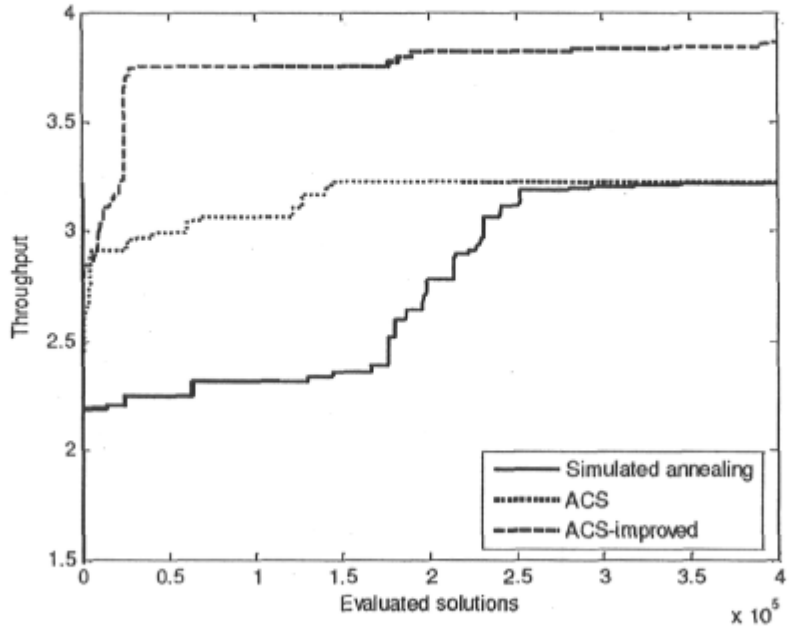

FIG. 4.16 – Convergence results for example 4

is the mean value obtained by either simulated annealing or the ACS without the improvement procedure.

Figure 4.17 presents the average errors obtained by the simulated annealing and ACS algorithms (relatively to ACS-I), and shows that the SA algorithm out-performs the ACS algorithm.

4.9 Conclusion

A new optimal design problem was formulated in this paper for production lines with unreliable machines and finite buffers. In the existing approaches, the only decision variable is the buffer size. The proposed approach extends these classical buffer space allocation problems by formulating a more complicated problem where the decision variables are buffer and type of machine, as well as the number of redundant machines. To estimate series-parallel production line performance, we propose to approximate each set of parallel machines as a single unreliable machine. An analytical decomposition approximation based on a fluid modelling can then be used to estimate the production rate of each possible configuration when running the optimization algorithm. Simulation results showed that the used approximate technique is very accurate. The simulations

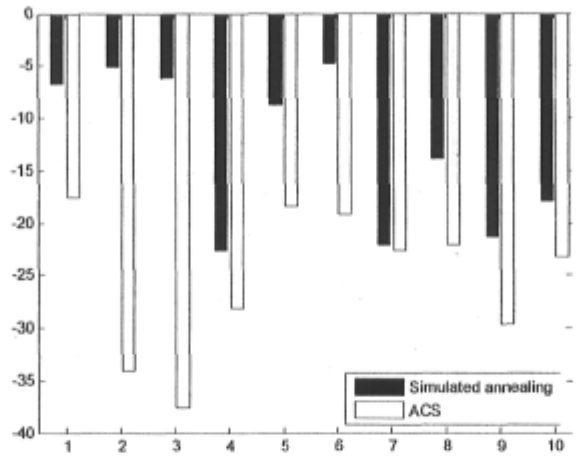

FIG. 4.17 – Comparing SA, ACS and ACS-I algorithms for $n = 20$

models were developed for three typical examples taken from the literature, and for 200 larger lines which we have randomly generated. As the formulated problem is a complicated combinatorial optimization one, an exhaustive examination of all possible solutions is not realistic, considering reasonable time limitations. Because of this, we developed and compared three algorithms to solve the formulated problem : simulated annealing, ACS alone, and ACS coupled with an improvement procedure. We have found that the SA algorithm generally out-performs the ACS algorithm. Nevertheless, when ACS is coupled with the improvement procedure, it generally yields better results than SA.

The comparison to others meta-heuristics (such as Tabu search, GRASP, Iterated Local Search, Variable Neighbourhood Search and Genetic Algorithms) is a perspective to investigate in the future by using the benchmarks proposed in this paper. Our current work concerns also the integration of maintenance aspects in the optimal design of buffered series-parallel production systems and other complex structures encountered in manufacturing systems.

4.10 Appendix

		Version 1	Version 2	Version 3	Version 4
Machine 1	μ, λ, P	0.0735, 0.0121, 2.2238	0.1470, 0.0289, 2.3338	0.0438, 0.0075, 2.3764	0.0392, 0.0068, 2.4643
	Cost($)	5	10	13	15
Machine 2	μ, λ, P	0.1557, 0.0530, 2.027	0.1588, 0.0446, 2.0396	0.1495, 0.0375, 2.1153	0.0358, 0.0082, 2.1273
	Cost($)	10	15	20	25
Machine 3	μ, λ, P	0.0521, 0.0053, 2.1849	0.0447, 0.0051, 2.2222	0.1383, 0.0205, 2.2924	0.0523, 0.0062, 2.3265
	Cost($)	10	12	15	20
Machine 4	μ, λ, P	0.0336, 0.0023, 2.1568	0.0916, 0.0037, 2.3087	0.1007, 0.0044, 2.3541	0.1074, 0.0055, 2.4084
	Cost($)	9	15	23	30

TAB. 4.7 – Machines data for example 1

		Version 1	Version 2	Version 3
Buffer 1	Capacity, Cost($)	30, 5	40, 8	55, 15
Buffer 2	Capacity, Cost($)	45,10	60, 20	65, 25
Buffer 3	Capacity, Cost($)	35, 5	50, 10	60, 18

TAB. 4.8 – Buffers data for example 1

		Version 1	Version 2	Version 3	Version 4	Version 5
Machine 1	μ, λ, P	0.2645, 0.0438, 3.1145	0.1544, 0.0268, 3.3136	0.2468, 0.0433, 3.3426	0.1593, 0.0269, 3.3977	0.3688, 0.0622, 3.5288
	Cost($)	5	7	10	12	13
Machine 2	μ, λ, P	0.2085, 0.0303, 2.9765	0.2171, 0.0283, 3.1229	0.3707, 0.0522, 3.4287	0.195, 0.0315, 3.5041	0.2525, 0.0339, 3.5389
	Cost($)	7	10	12	13	14
Machine 3	μ, λ, P	0.2351, 0.1242, 2.5327	0.2691, 0.0861, 2.783	0.1953, 0.0574, 2.9909	0.2192, 0.0484, 3.07	0.199, 0.0547, 3.2178
	Cost($)	10	11	12	15	17
Machine 4	μ, λ, P	0.1528, 0.0289, 2.9659	0.2796, 0.0535, 3.0089	0.2921, 0.0617, 3.2554	0.3878, 0.0798, 3.3826	0.1809, 0.0318, 3.398
	Cost($)	5	8	9	10	14
Machine 5	μ, λ, P	0.2314, 0.0242, 3.1254	0.2567, 0.0254, 3.1522	0.2103, 0.0175, 3.4252	0.1982, 0.021, 3.4279	0.1957, 0.0155, 3.5828
	Cost($)	20	22	23	25	26
Machine 6	μ, λ, P	0.1599, 0.0108, 3.2422	0.3256, 0.0221, 3.292	0.234, 0.018, 3.481	0.218, 0.0148, 3.5796	0.2695, 0.0196, 3.629
	Cost($)	20	22	23	25	26
Machine 7	μ, λ, P	0.2612, 0.0323, 3.0859	0.1718, 0.0217, 3.3243	0.1827, 0.0221, 3.3351	0.1552, 0.0172, 3.6137	0.3862, 0.047, 3.6151
	Cost($)	10	13	15	16	17
Machine 8	μ, λ, P	0.2287, 0.0108, 3.6734	0.3794, 0.015, 3.7013	0.1845, 0.0072, 3.818	0.2565, 0.0088, 3.8469	0.2403, 0.0096, 3.9015
	Cost($)	10	11	12	14	15
Machine 9	μ, λ, P	0.2494, 0.0153, 3.4925	0.2221, 0.0115, 3.5739	0.3126, 0.0198, 3.6441	0.1941, 0.0113, 3.7465	0.2547, 0.0126, 3.8411
	Cost($)	5	6	8	10	11
Machine 10	μ, λ, P	0.2285, 0.0051, 3.4271	0.3272, 0.0074, 3.6841	0.3721, 0.0124, 3.7248	0.1863, 0.0051, 3.828	0.3115, 0.0097, 3.9345
	Cost($)	15	16	18	20	21

TAB. 4.9 – Machines data for example 2

		Version 1	Version 2	Version 3	Version 4
Buffer 1	Capacity, Cost($)	40, 5	55, 8	70, 14	80, 20
Buffer 2	Capacity, Cost($)	30, 5	40, 8	50, 14	65, 20
Buffer 3	Capacity, Cost($)	30, 7	40, 10	45, 15	60, 18
Buffer 4	Capacity, Cost($)	45, 10	55, 15	60, 19	70, 23
Buffer 5	Capacity, Cost($)	35, 12	50, 15	67, 20	70, 30
Buffer 6	Capacity, Cost($)	40, 10	50, 15	65, 19	70, 23
Buffer 7	Capacity, Cost($)	50, 5	65, 8	75, 14	85, 20
Buffer 8	Capacity, Cost($)	30, 15	55, 20	65, 24	80, 28
Buffer 9	Capacity, Cost($)	30, 10	35, 15	40, 20	45, 25

TAB. 4.10 – Buffers data for example 2

		Version 1	Version 2	Version 3	Version 4	Version 5
Machine 1	μ, λ, P	0.2304, 0.1166, 2.6313	0.1973, 0.0972, 2.676	0.2139, 0.1336, 2.7256	0.0714, 0.0447, 2.8628	0.106, 0.0448, 2.9109
	Cost($)	5	7	10	12	13
Machine 2	μ, λ, P	0.1898, 0.0507, 3.1064	0.0923, 0.0205, 3.5717	0.1508, 0.0344, 3.7054	0.0792, 0.0204, 3.7109	0.0912, 0.019, 3.7525
	Cost($)	7	10	12	13	14
Machine 3	μ, λ, P	0.17, 0.025, 3.4031	0.0788, 0.0118, 3.4336	0.2261, 0.0327, 3.5206	0.0609, 0.0087, 3.8141	0.0615, 0.0092, 4.0129
	Cost($)	10	11	12	15	17
Machine 4	μ, λ, P	0.1093, 0.0404, 2.81	0.0682, 0.0222, 2.9181	0.1566, 0.0444, 3.0724	0.1804, 0.0562, 3.4688	0.1919, 0.0516, 3.667
	Cost($)	5	8	9	10	14
Machine 5	μ, λ, P	0.105, 0.0202, 3.4092	0.2216, 0.0352, 3.4678	0.1866, 0.0309, 3.6719	0.0751, 0.0113, 3.9665	0.0801, 0.0126, 3.9949
	Cost($)	20	22	23	25	26
Machine 6	μ, λ, P	0.1022, 0.0103, 3.5011	0.0719, 0.0073, 3.6115	0.1238, 0.0127, 3.652	0.0806, 0.0085, 3.6843	0.1755, 0.019, 4.243
	Cost($)	20	22	23	25	26
Machine 7	μ, λ, P	0.2373, 0.0236, 3.6365	0.1379, 0.0134, 4.1087	0.1587, 0.0156, 4.2079	0.1497, 0.0143, 4.212	0.1063, 0.0103, 4.2268
	Cost($)	10	13	15	16	17
Machine 8	μ, λ, P	0.0804, 0.0091, 3.4941	0.1803, 0.0206, 3.5102	0.1389, 0.0188, 3.6295	0.0767, 0.0089, 3.7977	0.1837, 0.0253, 4.0229
	Cost($)	10	11	12	14	15
Machine 9	μ, λ, P	0.1958, 0.0157, 3.6803	0.158, 0.0112, 3.8792	0.1185, 0.0094, 4.006	0.0979, 0.0077, 4.1912	0.0717, 0.0054, 4.2743
	Cost($)	5	6	8	10	11
Machine 10	μ, λ, P	0.166, 0.0147, 4.0969	0.0918, 0.0083, 4.1199	0.0975, 0.0082, 4.1955	0.0949, 0.0086, 4.2948	0.1807, 0.0148, 4.313
	Cost($)	15	16	18	20	21
Machine 11	μ, λ, P	0.1157, 0.008, 3.7438	0.1402, 0.0095, 3.8245	0.087, 0.0059, 3.9554	0.0641, 0.0045, 3.9678	0.0964, 0.0065, 4.0083
	Cost($)	8	13	15	16	17
Machine 12	μ, λ, P	0.079, 0.0013, 3.7965	0.0952, 0.0023, 4.019	0.2368, 0.0047, 4.2889	0.1972, 0.0044, 4.5438	0.0787, 0.0015, 4.5666
	Cost($)	9	11	12	15	17
Machine 13	μ, λ, P	0.1524, 0.1524, 1.9293	0.1309, 0.1309, 2.0227	0.1779, 0.1779, 2.1252	0.0762, 0.0762, 2.149	0.0983, 0.0983, 2.1636
	Cost($)	5	7	10	12	13
Machine 14	μ, λ, P	0.1997, 0.0077, 3.8336	0.0854, 0.0035, 4.1152	0.0791, 0.0034, 4.1519	0.0972, 0.0032, 4.1718	0.0688, 0.0019, 4.2924
	Cost($)	10	12	13	15	16
Machine 15	μ, λ, P	0.1573, 0.0099, 3.6318	0.0904, 0.0053, 3.7954	0.1232, 0.008, 4.0276	0.0637, 0.0039, 4.2929	0.1181, 0.0066, 4.3657
	Cost($)	5	8	10	12	15

TAB. 4.11 – Machines data for example 3

		Version 1	Version 2	Version 3	Version 4
Buffer 1	Capacity, Cost($)	40, 5	55, 8	70, 14	80, 20
Buffer 2	Capacity, Cost($)	30, 5	40, 8	50, 14	65, 20
Buffer 3	Capacity, Cost($)	30, 7	40, 10	45, 15	60, 18
Buffer 4	Capacity, Cost($)	45, 10	55, 15	60, 19	70, 23
Buffer 5	Capacity, Cost($)	35, 12	50, 15	67, 20	70, 30
Buffer 6	Capacity, Cost($)	40, 10	50, 15	65, 19	70, 23
Buffer 7	Capacity, Cost($)	50, 5	65, 8	75, 14	85, 20
Buffer 8	Capacity, Cost($)	30, 15	55, 20	65, 24	80, 28
Buffer 9	Capacity, Cost($)	30, 10	35, 15	40, 20	45, 25
Buffer 10	Capacity, Cost($)	30, 7	40, 10	45, 15	60, 18
Buffer 11	Capacity, Cost($)	35, 10	55, 15	60, 19	70, 23
Buffer 12	Capacity, Cost($)	30, 12	40, 15	67, 20	70, 30
Buffer 13	Capacity, Cost($)	40, 10	50, 15	65, 19	70, 23
Buffer 14	Capacity, Cost($)	40, 5	55, 8	65, 14	85, 20

TAB. 4.12 – Buffers data for example 3

		Version 1	Version 2	Version 3	Version 4	Version 5
Machine 1	μ, λ, P	0.0185, 0.0113, 1.7486	0.0813, 0.0115, 2.152	0.0931, 0.018, 2.2771	0.0353, 0.0072, 2.3781	0.0245, 0.0024, 2.6762
	Cost($)	5	7	10	12	13
Machine 2	μ, λ, P	0.0371, 0.0178, 1.9237	0.0202, 0.0042, 2.0775	0.0295, 0.0040, 2.3179	0.0303, 0.0019, 2.3961	0.0189, 0.0006, 2.7135
	Cost($)	7	10	12	13	14
Machine 3	μ, λ, P	0.0382, 0.0114, 1.8927	0.1223, 0.0204, 2.2257	0.128, 0.0098, 2.2937	0.0297, 0.0042, 2.2937	0.0186, 0.001, 2.8012
	Cost($)	10	11	12	15	17
Machine 4	μ, λ, P	0.0823, 0.0254, 1.9109	0.0585, 0.0226, 2.1123	0.0231, 0.001, 2.4097	0.0219, 0.0032, 2.4536	0.1037, 0.0205, 2.4773
	Cost($)	5	8	9	10	14
Machine 5	μ, λ, P	0.068, 0.0257, 1.8233	0.076, 0.0117, 2.1325	0.0611, 0.0022, 2.3506	0.08, 0.0134, 2.4409	0.1092, 0.0026, 2.7139
	Cost($)	20	22	23	25	26
Machine 6	μ, λ, P	0.0295, 0.0044, 2.2169	0.035, 0.0046, 2.5357	0.04804, 0.0068, 2.5703	0.033, 0.001, 2.7065	0.0354, 0.0012, 2.8585
	Cost($)	20	22	23	25	26
Machine 7	μ, λ, P	0.0353, 0.0139, 1.8043	0.03, 0.0029, 2.2432	0.1072, 0.0044, 2.3552	0.0294, 0.0006, 2.3913	0.0285, 0.0026, 2.5494
	Cost($)	10	13	15	16	17
Machine 8	μ, λ, P	0.0994, 0.0312, 1.8479	0.0359, 0.0072, 2.2335	0.0518, 0.0052, 2.2525	0.0285, 0.0007, 2.5563	0.0226, 0.0012, 2.7477
	Cost($)	10	11	12	14	15
Machine 9	μ, λ, P	0.0207, 0.0045, 2.1397	0.0988, 0.0286, 2.2173	0.0179, 0.0029, 2.3142	0.0891, 0.0126, 2.5253	0.0212, 0.0008, 2.5373
	Cost($)	5	6	8	10	11
Machine 10	μ, λ, P	0.094, 0.0142, 2.1576	0.1087, 0.013, 2.2653	0.0588, 0.0098, 2.4437	0.0931, 0.0058, 2.5515	0.0225, 0.0014, 2.6952
	Cost($)	15	16	18	20	21
Machine 11	μ, λ, P	0.0377, 0.0053, 2.152	0.1138, 0.0111, 2.3944	0.1134, 0.0107, 2.437	0.0216, 0.0012, 2.6218	0.1068, 0.0135, 2.6254
	Cost($)	8	13	15	16	17
Machine 12	μ, λ, P	0.0181, 0.0078, 1.8908	0.1131, 0.0068, 2.3775	0.0182, 0.0025, 2.4526	0.0572, 0.0042, 2.4693	0.1201, 0.0086, 2.4944
	Cost($)	9	11	12	15	17
Machine 13	μ, λ, P	0.0612, 0.006, 2.3013	0.0612, 0.0027, 2.5733	0.0594, 0.0055, 2.6149	0.0321, 0.0021, 2.6493	0.1287, 0.0065, 2.6888
	Cost($)	5	7	10	12	13
Machine 14	μ, λ, P	0.0281, 0.0123, 1.8713	0.0317, 0.0029, 2.3124	0.0192, 0.0036, 2.3596	0.0613, 0.0073, 2.6254	0.0764, 0.0072, 2.688
	Cost($)	10	12	13	15	16
Machine 15	μ, λ, P	0.0279, 0.0071, 2.06	0.0296, 0.0054, 2.1132	0.1134, 0.0182, 2.3418	0.1072, 0.0046, 2.5759	0.0311, 0.0012, 2.7956
	Cost($)	5	8	10	12	15
Machine 16	μ, λ, P	0.0307, 0.0034, 2.3659	0.0339, 0.0035, 2.5748	0.1291, 0.011, 2.6265	0.017, 0.0012, 2.6318	0.1279, 0.0044, 2.7025
	Cost($)	5	8	9	10	14
Machine 17	μ, λ, P	0.0888, 0.0449, 1.7617	0.0831, 0.0135, 2.3343	0.0346, 0.0029, 2.3881	0.0248, 0.0017, 2.3942	0.0175, 0.0005, 2.8557
	Cost($)	20	22	23	25	26
Machine 18	μ, λ, P	0.0316, 0.0097, 1.9782	0.03, 0.0022, 2.3827	0.0659, 0.0053, 2.5769	0.0704, 0.0078, 2.6572	0.0215, 0.0013, 2.775
	Cost($)	20	22	23	25	26
Machine 19	μ, λ, P	0.0615, 0.0183, 2.2777	0.0334, 0.0076, 2.3789	0.0172, 0.0026, 2.3951	0.123, 0.0084, 2.6452	0.032, 0.0009, 2.6673
	Cost($)	10	13	15	16	17
Machine 20	μ, λ, P	0.0244, 0.0073, 1.9532	0.0715, 0.0075, 2.4901	0.1182, 0.0061, 2.6246	0.0291, 0.0019, 2.644	0.0719, 0.0057, 2.6495
	Cost($)	10	11	12	14	15

TAB. 4.13 – Machines data for example 4

		Version 1	Version 2	Version 3	Version 4
Buffer 1	Capacity, Cost($)	40, 5	55, 8	70, 14	80, 20
Buffer 2	Capacity, Cost($)	30, 5	40, 8	50, 10	65, 13
Buffer 3	Capacity, Cost($)	30, 7	40, 10	45, 11	60, 15
Buffer 4	Capacity, Cost($)	45, 10	55, 12	60, 15	70, 20
Buffer 5	Capacity, Cost($)	35, 12	50, 15	67, 20	70,30
Buffer 6	Capacity, Cost($)	40, 10	50, 11	65, 13	70, 14
Buffer 7	Capacity, Cost($)	50, 5	65, 8	75, 10	85, 13
Buffer 8	Capacity, Cost($)	30, 17	55, 20	65, 24	80, 28
Buffer 9	Capacity, Cost($)	30, 10	35, 12	40, 15	45, 16
Buffer 10	Capacity, Cost($)	30, 7	40, 10	45, 15	60, 18
Buffer 11	Capacity, Cost($)	35, 10	55, 12	60, 15	70, 18
Buffer 12	Capacity, Cost($)	30, 12	40, 15	67, 20	70, 23
Buffer 13	Capacity, Cost($)	40, 10	50, 15	65, 19	70, 23
Buffer 14	Capacity, Cost($)	40, 5	55, 8	65, 13	85, 17
Buffer 15	Capacity, Cost($)	50, 5	65, 8	75, 12	85, 15
Buffer 16	Capacity, Cost($)	30, 15	55, 16	65, 18	80, 21
Buffer 17	Capacity, Cost($)	30, 10	35, 15	40, 20	45, 25
Buffer 18	Capacity, Cost($)	30, 7	40, 10	45, 12	60, 15
Buffer 19	Capacity, Cost($)	35, 10	55, 15	60, 19	70, 23

TAB. 4.14 – Buffers data for example 4

Bibliographie

[1] Almotawa S., Savar M. and Al-Rashdan K. *Optimum machine selection in multistage manufacturing systems.*International Journal of Production Research, 2005, 43(6), 1109-1126.

[2] Burman M.H. *New Results in Flow Line Analysis.* Ph. D. Thesis, MIT, Cambridge MA., 1995.

[3] Buzacott J.A. *Automatic transfer lines with buffer stocks.* International Journal of Production Research, 1967, 5(3), 182-200.

[4] Buzacott J.A. *Prediction of efficiency of production systems without internal storage.* International Journal of production Research, 1968, 6(3), 173-188.

[5] Chern MS. *On the computational complexity of reliability redundancy allocation in a series system.* Operations Research Letter 1992 ;11 :309–15.

[6] Choong Y.F. and Gershwin S.B. *A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times.* IIE Transactions, 1989, 19, 150–159.

[7] Ciprut P, Hongler M-O and Salama Y. *On the variance of the production output of transfer lines.* IEEE Transactions on Robotics and Automation, 1999, 15(1), 33-43.

[8] Dallery Y, David R and Xie X.L. *An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers.* IIE transactions, 1988, 20(3), 280-283.

[9] Dallery Y and Le Bihan H. *Homogenisation techniques for the analysis of production lines with unreliable machines having different speeds.* European Journal of Control, 1997, 3, 200-215.

[10] Dallery Y and Gershwin S.B. *Manufacturing flow line systems : a review of models and analytical results.* Queuing Systems theory and Applications, Special Issue on Queuing Models of Manufacturing Systems, 1992, 12(1-2), 3-94.

[11] Daskalaki S. and Smith J.M. *Combining routing and buffer allocation problems in series-parallel queuing networks.* Annals of Operations Research, 2004, 125, 47-68.

[12] Dorigo M, Maniezzo V and Colorni A. *The Ant System : Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man and Cybernetics- Part B, 1996, 26(1), 1-13.

[13] Dorigo M and Gambardella LM. *Ant colony system : a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation, 1997, 1(1), 53–66.

[14] Dorigo M. and Stutzle T. *The ant colony optimization metaheuristic : algorithms, applications and advances.* Handbook of Metaheuristics, F. Glover and G. Kochenberger, 2001.

[15] Gershwin S.B. *An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking.* Operations Research, 1987, 35, 291-305.

[16] Gershwin S.B. *Representation analysis of transfer lines with machines that have different processing rates.* Ann. Oper. Res, 1987, 9, 511-530.

[17] Gershwin S.B. *An efficient decomposition algorithm for unreliable tandem queueing systems with finite buffers.* In H. G. Perros and T. Altiok, (ed), Queueing Networks with Blocking (North Holland), 1989, 127-146.

[18] Gershwin S.B. *Manufacturing systems engineering.* Prentice-Hall, 1994.

[19] Gershwin S.B and Schor J.E. *Efficient algorithms for buffer space allocation.* Annals of Operations Research, 2000, 93, 117-144.

[20] Hongler M-O and Salama Y. *Versus discrete flow of parts in a production dipole : exact transient analysis.* International conference on Emerging Technol. and Factory Automation (ETFA'95), 1995, Paris, France.

[21] Huang M-G, Chang P-L and Chou Y-C. *Buffer allocation in flow-shop-type production systems with general arrival and service patterns.* Computers & Operations Research, 2002, 29, 103-121.

[22] Kirckpatrick S, Gerlatt C.D Jr and Vecchi M.P. *Optimization by simulated annealing.* Science, 1983, 220, 671-680.

[23] Koulmas C., Antony S.R and John R. *A survey of simulated annealing applications to operations research problems.* Omega International Journal of Management Science, 1994, 22, 41-56.

[24] Le Bihan, H and Dallery Y. *Homogenization techniques for the analysis of production lines with unreliable machines having different speeds.* Eur. J. Contr., 1997, 3 (3), 200-215.

[25] Levitin G and Meizin L. *Structure optimization for continuous production systems with buffers under reliability constraints.* International Journal of Production Economics, 2001, 70, 77-87.

[26] Liu X-G and Buzacott J.A. *Approximate models of assembly systems with finite banks.* European Journal of Operational Research, 1990, 45, 145-154.

[27] Nahas, N, Ait-Kadi, D and Nourelfath M. *A new approach for buffer allocation in unreliable production lines.* International Journal of Production Economics, 2006, 103(2), 873-881.

[28] Nahas, N. and Nourelfath M. *Ant system for reliability optimization of series system with multiple-choice and budget constraints.* Reliability Engineering and System Safety, 2005, 87, 1-12.

[29] Nourelfath M, Nahas N and Ait-Kadi. *Optimal design of series production lines with unreliable machines and finite buffers.* Journal of Quality and Maintenance Engineering, 2005, 11(2), 121-138.

[30] Pincus M. *A monte carlo method for the approximate solution of certain types of constrained optimization problems.* Oper. Rese., 1970, 18, 1225-1228.

[31] Shi L and Men S. *Optimal buffer allocation in production lines.* IIE Transactions, 2003, 35, 1-10.

[32] Smith J.M and Cruz F.R.B. *The buffer allocation problem for general finite buffer queueing networks.* IIE Transactions, 2005, 37, 343-365.

[33] Spinellis D, Papadopoulos C.T. *A simulated annealing approach for buffer allocation in reliable production lines.* Annals of Operations Research, 2000, 93, 373-384.

[34] Tempelmeier H. *Practical considerations in the optimization of flow production systems.* International Journal of Production Research, 2003, 41(1), 149-170.

[35] Tripathi A.K, Tiwari M.K and Chan F.T.S. *Multi-agent-based approach to solve part selection and task allocation problem in flexible manufacturing systems.* International Journal of Production Research, 2005, 43(7), 1313-1335.

Chapitre 5

Conclusion générale

Dans cette thèse, le problème de conception optimale des systèmes de production dont les processeurs sont assujettis à des défaillances aléatoires a été traité. Trois types de systèmes ont été étudiés : lignes de productions sans stocks tampons, lignes de production séries et séries-parallèles avec stocks tampons. Pour chacun de ces systèmes, la résolution du problème de conception a nécessité l'élaboration de deux principaux outils :

- une méthode d'évaluation de la performance (fiabilité ou taux de production).
- une méthode d'optimisation qui utilise l'outil d'évaluation pour le calcul de la fonction objectif.

Pour les systèmes sans stocks, l'évaluation de la fiabilité a été largement abordée dans la littérature. En ce qui concerne les systèmes séries avec stocks, les travaux existants sont généralement basés sur la méthode de décomposition ou d'agrégation. Les lignes séries-parallèles avec stocks tampons sont plus complexes à étudier analytiquement. Dans la littérature actuelle, peu de travaux ont été dédiés à cette problématique. Nous avons contribué à ce sujet en proposant une nouvelle méthode d'évaluation de performance de ces lignes qui donne des résultats très précis en un temps très court. Cette méthode est basée sur la combinaison de trois approximations (i.e. aggrégation des machines parallèle, homogénéisation de la ligne et l'utilisation d'une méthode de type décomposition). La validation de cette méthode a nécessité une étude comparative détaillée en se basant sur la simulation du système. Il a fallu donc simuler des lignes de production séries-parallèles en s'approchant le plus possible de la réalité industrielle et en comparant les résultats obtenus (performance) avec ceux obtenus avec la méthode approximative proposée. Le défit rencontré était de générer un grand nombre de données spécifiques aux machines et aux stocks tampons (taux de pannes, taux de réparations, cadence des machines et les capacités des stocks tampons) d'une façon aléatoire. La génération aléatoire de telles données pourra engendrer des problèmes de goulots d'étranglements. Pour y remédier, nous nous sommes inspirés de la procédure développée par Mitchell H. Burman dans ses travaux de thèse en 1995. Cette procédure permet de générer des lignes de production réalistes. Ces données ont été utilisées pour valider la méthode approximative d'évaluation de performance proposée dans cette thèse. Nous avons donc montré que la méthode proposée présente des résultats assez précis lorsque comparés à ceux des simulations.

En ce qui concerne les outils d'optimisation, les principaux défis étaient de concevoir des algorithmes robustes et efficaces pour chacun des problèmes et d'améliorer les résultats obtenus par les approches existantes. La conception d'un algorithme à base de métaheuristiques est une tâche difficile qui nécessite à la fois une connaissance approfondie du problème étudié et du fonctionnement intrinsèque de cette ou ces métaheuristique(s). Dans le cas d'un algorithme de type colonies de fourmis, chaque problème peut être représenté par plusieurs types de graphes, et le choix du "meilleur"

graphe est une tâche qui est loin d'être systématique. De plus, un choix judicieux de l'information heuristique est nécessaire pour la conception d'un algorithme à colonies de fourmis efficace. Dans le cas d'un algorithme de recherche locale, tel que le grand déluge étendu ou le recuit simulé, le défi réside dans la construction d'un type de voisinage simple et permettant une bonne exploration de l'espace des solutions. Par ailleurs, l'hybridation de deux métaheuristiques nécessite la mise en place d'une interaction adéquate entre les deux algorithmes, ajoutant ainsi à la complexité globale de la conception d'un nouvel algorithme à base de métaheuristiques.

Notons enfin que la conception d'un algorithme à base de métaheuristiques constitue un processus qui évolue de l'étude de cas simples à celle de cas plus complexes. En effet, des résultats préliminaires ont été tout d'abord obtenus à l'aide de métaheuristiques de base (algorithme à colonies de fourmis, recuit simulé et algorithme du grand déluge étendu). Pour le problème de conception des lignes de production séries-parallèles sans stocks tampons, le défi était de taille étant donné que les approches présentées dans la littérature étaient très efficaces et robustes. Ceci vient du fait que le problème a été largement traité tout au long des quatre dernières décennies. Ce constat nous a poussés à concevoir un algorithme d'optimisation hybride incluant deux approches (algorithme à colonies de fourmis et l'algorithme du grand déluge étendu) pour se comparer aux résultats publiés. On s'est aperçu que dans certains cas, les résultats obtenus par hybridation sont meilleurs. Pour les problèmes de conception des lignes séries et séries-parallèles avec stocks tampons, la principale difficulté rencontrée était le peu d'exemples-tests traités dans la littérature. Par conséquent, il a été donc nécessaire de générer d'une façon aléatoire des exemples de grande taille et la conception d'au moins deux algorithmes pour chacun des deux problèmes afin de mener à bien une étude comparative.

D'une manière générale, cette étude a montré que les métaheuristiques comme l'algorithme à colonies de fourmis performent mieux lorsqu'elles sont couplées avec une procédure d'amélioration locale qui peut être soit une autre métaheuristique ou bien un simple algorithme d'amélioration.

Dans ce qui suit, une récapitulation des contributions des trois articles de la thèse et des perspectives de recherche sont présentées.

La principale contribution du premier article réside dans l'élaboration d'un algorithme hybridant deux métaheuristiques, les colonies de fourmis et le grand déluge étendu, pour résoudre le problème d'allocation de la redondance. Les résultats obtenus pour les différents problèmes tests dans la littérature montrent que l'efficacité et la robustesse de l'algorithme proposé en comparaison avec les approches existantes.

La principale contribution du second article est l'élaboration d'un algorithme à base de la métaheuristique du grand déluge étendu pour résoudre efficacement le problème d'allocation optimale des stocks. Une étude comparative avec le recuit simulé sur des exemples générés aléatoirement montre l'efficacité de cet algorithme en termes de temps de convergence et de qualité des solutions.

Le troisième article développe plusieurs contributions originales :
- Un nouveau problème de conception optimale est formulé ;
- Une nouvelle méthode analytique est proposée pour l'évaluation du taux de production d'une ligne série-parallèle. Cette méthode est validée par simulation.
- Deux algorithmes à base de métaheuristiques sont proposés. Une comparaison sur des problèmes tests générés montre que l'algorithme à colonies de fourmis couplé à une procédure d'amélioration est meilleur que l'algorithme à base du recuit simulé.

Les divers modèles et algorithmes proposés dans la thèse représentent des contributions vers la conception optimale des systèmes de production. Ces contributions originales ont été publiées dans trois revues de génie industriel différentes et qui sont parmi les meilleures dans les domaines de la fiabilité et de la production.

Les perspectives de ces travaux sont :
- Le développement d'une nouvelle méthode d'évaluation de performance des systèmes séries parallèles pouvant être à base des méthodes de décomposition ou d'agrégation.
- La résolution du problème de conception optimale des lignes de production en considérant, dans ce cas, à la fois l'allocation optimale des stocks (comme c'est le cas du chapitre 4) et le choix des technologies des machines.
- La calibration automatique des paramètres des différents algorithmes à base de métaheuristiques proposés.
- L'intégration des aspects de la maintenance corrective et de la maintenance préventive dans la conception optimale des lignes de production avec stocks tampons.
- L'exploration d'autres métaheuristiques, telles que la recherche avec des tabous et les algorithmes génétiques, et d'autres types d'hybridation pour résoudre les différents problèmes de conception optimale étudiés dans cette thèse.
- L'extension de la méthodologie proposée vers d'autres types de systèmes plus complexes, tels que les systèmes opérant en structure réseau et les systèmes multi-états.