ISSN : 0953-7287 (imprimé) 1366-5871 (en ligne) Page d'accueil de la revue : https://www.tandfonline.com/loi/tppc20

Note technique : Nouveaux résultats pour le problème de dimensionnement des lots avec capacité avec décisions en heures supplémentaires et temps de configuration

Linet Özdamar, Şevket İlker Bilbil & Marie-Claude Portmann

Pour citer cet article : Linet Özdamar , Şevket İlker Bilbil & Marie-Claude Portmann (2002)
Note technique : Nouveaux résultats pour le problème de dimensionnement des lots avec capacité avec décisions en heures supplémentaires et temps de configuration, , 13:1, 2-10, DOI : 10.1080/09537280110049272

Pour faire le lien vers cet article : https://doi.org/10.1080/09537280110049272

Publié en ligne : 15 novembre 2010.

PLANIFICATION ET CONTRÔLE DE LA PRODUCTION, 2002, VOL. 13, NON. 1, 2–10

Note technique : Nouveaux résultats pour le problème de dimensionnement des lots avec capacité avec décisions en heures supplémentaires et temps de configuration

LINET O'ZDAMAR, S¹ EVKET I_LKER BIRBIL et
MARIE-CLAUDE PORTMANN

Mots clés lot sizing, algorithme génétique, heuristique

Abstrait. Le problème de dimensionnement des lots capacités avec heures supplémentaires et temps d'installation (CLSPOS) consiste à planifier les tailles de lots de plusieurs familles sur un horizon de planification dans le but de minimiser les heures supplémentaires et les coûts de détention des stocks. Chaque fois que la taille de lot d'un article est positive, la capacité est consommée par une configuration. La capacité est limitée et comprend la capacité de temps régulier ainsi que les heures supplémentaires. On suppose que les configurations n'entraînent pas de coûts autres que la capacité de production perdue et, par conséquent, les configurations contribuent implicitement aux coûts totaux via les coûts des heures supplémentaires chaque fois que des goulots d'étranglement de capacité se produisent. Le CLSPOS est plus compliqué que le problème standard de dimensionnement de lot capacité (CLSP) qui implique des coûts de configuration explicites, aucune configuration consommatrice de capacité et uniquement une capacité de temps régulière. Ici est décrit un algorithme génétique (GATA) intégré avec la recherche tabou (TS) et le recuit simulé (SA) pour résoudre CLSPOS. GATA intègre les puissantes caractéristiques des trois algorithmes de recherche, GAs, TS et SA. Les résultats sont comparés à ceux rapportés dans une étude précédente et démontrent que GATA surpasse les autres heuristiques.

1. Introduction

Le problème de dimensionnement de lot capacité à une étape (CLSP) est un problème NP-difficile (Bitran et Yanasse, 1982) dans la planification de la production. Dans le CLSP, la demande est déterministe mais variable dans le temps, l'horizon de planification se compose d'un nombre fini de périodes discrètes, la capacité est limitée et aucune commande en souffrance n'est autorisée. Une solution au CLSP consiste en des tailles de lots d'articles qui satisfont la demande tout au long de l'horizon de planification sans dépasser la limite de capacité à chaque période. Chaque fois qu'un article est produit, un coût d'installation est engagé. La fonction objective minimise les coûts d'installation et d'inventaire. Les méthodes de solution heuristique proposées pour ce problème impliquent généralement une heuristique période par période (par exemple, Maes et Van Wassenhove 1986, Gu'nther 1987) où la taille des lots pour les familles est déterminée sur la base d'un critère d'économie de coût.

Chaque période, la demande future est satisfaite jusqu'à ce qu'aucun coût supplémentaire ne puisse être économisé ou que la capacité de cette période soit épuisée. Il existe également des approches heuristiques basées sur la programmation mathématique (par exemple, Trigeiro 1989, Cattrysse et al. 1990). Comme Trigeiro et al. (1989) remarquent que lorsque les temps de configuration sont inclus dans le modèle, le CLSP devient également NP-Complet dans le problème de faisabilité.

Le modèle proposé ici étend le CLSP à deux égards : (i) il suppose deux sources de capacité, régulières

Auteurs : Linet O'zdamar et SË evket _Ilker Birbil, Yeditepe University, Department of Systems Engineering, Istanbul, Turkey, E-mail : lozdamar@hotmail.com, et Maria-Claude Portmann, LORIA-INPL – Ecole des Mines de Nancy, France.

Linet O'zdamar est titulaire d'un BSc, d'un MSc et d'un PhD de BoDazicËi, Université, Département de Génie Industriel, Istanbul Turquie. Ses intérêts de recherche portent sur la planification hiérarchique de la production, l'ordonnancement de projets, les métaheuristiques, l'optimisation globale et la théorie floue de base. Elle est membre des groupes de travail de l'EURO sur la gestion de projet et l'ordonnancement, l'ingénierie industrielle et l'économie de la production, et sur la prise de décision distribuée. Elle a publié des articles connexes dans des revues telles que European Journal of Operations Research, International Journal of Production Research, IEEE Transactions on Systems, Man and Cybernetics, IEEE Transactions on Automatic Control, etc.

Planification et contrôle de la production ISSN 0953–7287 imprimé/ISSN 1366–5871 en ligne # 2002 Taylor & Francis Ltd http://www.tandf.co.uk/journals
DOI : 10.1080/0953728011004927 2

du temps et des heures supplémentaires, ce qui entraîne une perte d'homogénéité de la capacité, (ii) les installations consomment de la capacité et n'engendrent pas de coûts autres que la perte de capacité de production. Par conséquent, les coûts d'installation ne sont pas explicitement indiqués dans la fonction objectif, mais ils entraînent des coûts implicites lorsque des goulots d'étranglement de capacité se produisent et que la capacité d'heures supplémentaires est utilisée. Dans le problème qui en résulte, la fonction objectif consiste en la tenue des stocks et les coûts des heures supplémentaires tandis que les contraintes garantissent que la demande est satisfaite à chaque période et que la somme des capacités de temps régulier et d'heures supplémentaires n'est pas dépassée. Ce problème est appelé problème de dimensionnement de lot capacité avec décisions et temps de configuration des heures supplémentaires (CLSPOS). La formulation mathématique du CLSPOS est donnée dans le tableau 1. Diaby et al. (1992) développent une relaxation lagrangienne basée sur des limites de capacité desserrées et une optimisation des sous-gradients pour CLSPOS. Cependant, ils supposent que toutes les familles ont les mêmes délais de traitement.

Ici est développé un puissant algorithme génétique (GATA) intégré à une procédure de recherche locale (TSSA) incorporant des fonctionnalités de recherche taboue (TS) et de recuit simulé (SA). GATA implique un encodage direct du CLSPOS, et comme il est diµcile de maintenir la faisabilité de toutes les contraintes, il conduit généralement la recherche dans l'espace des solutions infaisables. Alors que GATA

Tableau 1. Formulation mathématique du problème de dimensionnement de lot capacité avec des décisions en temps supplémentaire et des temps de préparation.

Modèle P :

$$\text{X minimum } \sum_t \sum_j \text{Ijthj} \ddagger \text{X sur Ot}$$

$$\text{St: Ij;t¡1} \ddagger \text{Yjt ¡ Ijt ^ djt X} \qquad 8\ j;\ t$$

$$\ldots \text{pjYjt} \ddagger \text{wjtsj† µ Ct} \ddagger \text{Ot } 8\ t$$

$$\text{Ot µCOt \_} \qquad 8 \text{ tonnes}$$

$$\text{Yjt µ Mwjt} \qquad 8\ j;\ t$$

où djt = demande de l'article j dans la période t pj = temps de traitement de l'article j (hr) ov = coût des heures supplémentaires hj = coût de détention par unité de l'article j par période Ct = capacité horaire régulière dans la période t COt = capacité des heures supplémentaires dans période t Ijt = stock de l'article j détenu à la fin de la période t … Ijt 0† Yjt = taille du lot de production de l'article j pendant la période t …Yjt 0† Ot = capacité d'heures supplémentaires utilisée pendant la période t …Ot 0† sj = temps de réglage requis pour l'item j wjt = variable binaire (= 1, si l'item j est produit dans la période t ; = 0, sinon) M = un grand nombre

imite le processus évolutif, TSSA est activé de temps en temps en fonction de la diversité de la population actuelle. La TSSA réduit l'éventail des performances de la population en encourageant les mouvements qui rendent faisables les chromosomes sur lesquels elle agit. Ensuite, GATA l'étend en utilisant des opérateurs de croisement et de mutation qui peuvent détruire la faisabilité de la capacité. Les phases consécutives de contraction/expansion aboutissent à une convergence rapide vers la solution optimale. GATA bénéficie également de la migration et exécute une recherche simultanée sur des populations parallèles. Dans les résultats de calcul, les performances de GATA sont comparées à un GA impliquant un codage indirect du CLSPOS (IGA) développé dans une étude précédente (Ozdamar et Bozyel 2000) ainsi qu'à

d'autres méthodes citées dans cette dernière référence.

2. L'algorithme génétique - GATA

Les algorithmes génétiques (AG) sont de puissants outils de recherche souvent utilisés en optimisation. Des revues approfondies des AG sont données par Portmann (1996) et Alexandre et al. (1997). Les AG émulent la sélection naturelle et la diversité en appliquant des opérateurs de croisement, de reproduction et de mutation à une population de solutions. Une autre méthode de recherche probabiliste est le recuit simulé (SA) (Kirkpatrick et al. 1983) qui est une technique de recherche de solution unique où les pires solutions sont acceptées par un programme de refroidissement similaire à un processus de recuit. De cette manière, la recherche peut échapper aux optima locaux. La recherche taboue (TS) fonctionne également sur une seule solution, mais une nouvelle solution est générée en étudiant tous les voisins immédiats de la solution actuelle et le meilleur voisin devient la nouvelle solution actuelle même si elle ne s'améliore pas. Pour éviter de revisiter les mêmes solutions, une liste tabou garde une trace d'un certain nombre de mouvements précédents.

En tant que métaheuristique hybride où les trois approches de recherche sont fusionnées, la procédure GATA développée ici intègre bon nombre de leurs avantages.
GATA travaille sur plusieurs populations et migre les chromosomes en exécutant une opération croisée qui réalise un mélange interpopulationnel. Ainsi, la diversité de la population est accrue et la convergence prématurée est évitée. De plus, la liberté de visiter des espaces de solutions qui sont très loin de contenir la solution optimale est restreinte par une exécution occasionnelle de TSSA qui améliore les performances de la population et pousse les chromosomes vers des régions réalisables intéressantes.

Les procédures GATA et TSSA sont toutes deux capables d'effectuer la recherche dans des régions irréalisables ainsi que dans des régions réalisables. Les opérateurs de croisement et de mutation dans GATA ne préservent que la non négativité des niveaux d'inventaire et par conséquent, ce sont des opérateurs rapides. La capacité

les infaisabilités impliquées par un ensemble de tailles de lot sont pénalisées dans la fonction objectif proportionnellement à l'excédent de capacité total. Par conséquent, GATA effectue généralement la recherche dans l'espace irréalisable pendant les phases initiales de la recherche. Cette fonctionnalité contribue non seulement à l'eцcacité de calcul de GATA, mais ajoute également une diversité supplémentaire à la population.

GATA active la procédure TSSA lorsque la population est susceptible de rester bloquée dans la même zone de l'espace des solutions faisables/infaisables. La TSSA effectue une recherche locale sur des chromosomes choisis au hasard dans la population actuelle. Un chromosome sélectionné au hasard par TSSA est remplacé par la meilleure solution obtenue lors de la recherche locale. Bien sûr, il n'y a aucune garantie qu'à partir d'un chromosome infaisable, TSSA se retrouvera dans un chromosome faisable de capacité. Le nombre de chromosomes traités par TSSA est égal à PopSize*r, où PopSize est la taille de la population et r est une constante inférieure à un.

Une fois que la TSSA a terminé la recherche, les opérateurs GA prennent le relais pour poursuivre le processus d'évolution. Le

Tableau 2. Pseudocode pour la procédure GATA.

```
Procédure GATA ;

commencer pour i ^ 1 à PopNo
        génère des chromosomes PopSize ;
    répéter
        pour i ^ 1 jusqu'à PopNo
        faire pour j ^ 1 jusqu'à
            GenNo
                faire commencer si non(SA) puis reproduire ;
                SA :^ faux ;
                croisement;
                calculer PopRange ; si
                PopRange > ¬ alors mutation ; calculer
                PopRange ; si PopRange
                > ¶ alors commencer

                        Faire TSSA sur di¶érents chromosomes r*PopSize ;
                        SA :^ vrai ; fin;
                    fin;

        Émigrer;
    jusqu'à ce qu'une condition d'arrêt soit
vérifiée ; fin;

Dé nitions :

PopNo : nombre de populations parallèles GenNo :
nombre de générations sans migration PopSize : nombre de chromosomes
dans la population SA : booléen indiquant si la TSSA vient
d'être mise en place r : fraction de la population subissant la TSSA PopRange :
rapport de la fonction objectif minimale valeur au
maximum dans la population ¬; ¶ : constantes inférieures à un.
```

La procédure TSSA diversifie la population à l'instar de l'opérateur de mutation, car la meilleure solution obtenue à l'issue de la recherche di¶ère considérablement de la solution initiale. D'autre part, la valeur de la fonction objectif de la meilleure solution est au moins aussi bonne et très probablement bien inférieure à celle de la solution initiale.
Par conséquent, comme la TSSA opère sur un certain nombre de chromosomes dans la population, la quantité d'infaisabilité dans la population diminue. En termes de performances de la population, TSSA intensifie la recherche tout en ayant un e¶et de diversification sur l'espace des solutions couvert par la population, à condition que les chromosomes améliorés ne convergent pas vers un trop petit ensemble d'optimums locaux.

Un pseudocode pour GATA est donné dans le tableau 2. GATA commence par un nombre prédéterminé (PopNo) de populations initiales générées aléatoirement et procède avec chacune d'elles en parallèle en appliquant des phases GA/TSSA consécutives pendant un certain nombre de générations (GenNo). La procédure TSSA est activée lorsque la plage de performance de la population (PopRange), c'est-à-dire le rapport de la valeur minimale de la fonction objective parmi les chromosomes à la valeur maximale, atteint presque la valeur de un malgré le fait que l'opérateur de mutation vient d'être activé.
Ensuite, les parents de di¶érentes populations sont croisés et le rejeton inséré dans leurs populations parentes correspondantes (opérateur Migrer). Chaque population est retraitée indépendamment jusqu'à la prochaine migration.
La procédure continue jusqu'à ce qu'une condition d'arrêt soit vérifiée.

3. Encodage chromosomique dans GATA et population initiale

Le codage chromosomique dans GATA est direct (Portmann 1996) dans le sens où il se compose simplement des variables de taille de lot yjt indiquées dans le modèle mathématique. La valeur de la fonction objectif d'une solution représentée par un chromosome c est indiquée par objc .

Un chromosome dans la population initiale est généré en attribuant une taille de lot aléatoire à chaque élément de chaque période. Si une taille de lot attribuée au hasard ne satisfait pas la demande dans la période considérée, elle est satisfaite en augmentant aléatoirement les tailles de lot des périodes précédentes.
Ainsi, un chromosome est garanti d'avoir un inventaire non négatif, mais les exigences de capacité impliquées par les tailles de lot peuvent violer les limites.

4. Opérateurs GA : reproduction, croisement, mutation

L'opérateur de reproduction multiplie chaque chromo en proportion de sa valeur de fonction objectif. Le

La fonction de précision utilisée pour reproduire un chromosome est un polynôme du second degré de l'inverse de la valeur de la fonction objective du chromosome. La probabilité de reproduire un chromosome k donné est donnée par,

$$\text{problème} \hat{} \frac{\ldots 1=\text{objk}\dagger^{2}}{X \quad \ldots 1=\text{objk}\dagger^{2}}$$

k °population

Cependant, la reproduction n'est pas exécutée immédiatement après la mise en œuvre de la TSSA, de sorte que les blocs critiques de gènes dans les chromosomes non traités par la TSSA sont préservés dans la population, car les valeurs de fonction objective de ces chromosomes sont comparativement très grandes (en raison des pénalités d'infaisabilité) par rapport aux faisables traités par TSSA.

L'opérateur de croisement est un opérateur à deux points qui sélectionne au hasard deux parents dans la population et deux périodes, t1 et t2 , au hasard. yjt sont échangés entre les deux parents pour tous les t1 µ t µ t2 . L'opérateur de croisement peut détruire la faisabilité des contraintes d'équilibre des stocks. C'est-à-dire qu'il peut y avoir surproduction ou sous-production chez l'o¶printemps. Ces infaisabilités sont réparées en ajoutant/soustrayant les montants nécessaires à/des tailles de lot. L'opération de réparation s'effectue d'abord en calculant la demande cumulée pour chaque article, en partant de la première période jusqu'à la dernière et en vérifiant si les tailles de lot cumulées sont supérieures ou égales à la demande cumulée de chaque période. Dans le cas où cette dernière condition est violée pour un article, seul le montant minimal est ajouté à la taille du lot de la période actuellement considérée. Ce laissez-passer de réparation anticipée évite les commandes en attente. Dans la passe arrière, le stock de chaque article est vérifié au cours de la dernière période et si le stock est positif, le minimum entre la taille du lot de cet article et son niveau de stock est déduit de la taille du lot. Si le niveau de stock de la dernière période est toujours positif après déduction, alors les tailles de lots des périodes précédentes sont déduites de manière à retirer complètement le stock de la dernière période. La passe de réparation arrière évite une surproduction.

Un exemple clarifiera la fonction de réparation dans l'opérateur de croisement. Supposons que dans le premier chromosome parent, la taille du lot d'un élément soit (15, 35, 0, 70, 40) pour un horizon de planification de cinq périodes. La demande pour l'article est de (5, 40, 5, 25, 85) unités. Le deuxième parent a des tailles de lot de (5, 60, 20, 75, 0) unités. Lorsque t1 ˆ 2 et t2 ˆ 4, le ressort aura respectivement (15, 60, 20, 75, 40) unités et (5, 35, 0, 70, 0) unités. Le premier ressort a une surproduction de 50 unités et le second a une sous-production de 50 unités. La passe en avant identifie en seconde période une commande en souffrance de 5 unités en seconde période et la répare en ajoutant 5 unités à la taille du lot de la seconde période. De même, 5 et 40 unités de commandes en souffrance sont réparé dans les troisième et cinquième périodes. Les tailles de lot modifiées résultantes sont (5, 40, 5, 70, 40) unités. La passe en arrière répare l'inventaire positif du premier printemps de la dernière période, en soustrayant 40 unités de la taille du lot de la dernière période et 10 unités de la taille du lot de la quatrième période pour obtenir les tailles de lot (15, 60, 20, 65, 0).

Dans GATA, le nombre de descendants pour remplacer la population parente est donné par PopSize*PopRange. Ainsi, à mesure que la diversité de la population se perd, le nombre de nouveaux descendants augmente. Cependant, lorsque la diversité de la population est grande, PopRange peut parfois être trop petit pour que très peu de nouveaux rejetons soient créés. Pour éviter cette dernière situation, une limite inférieure de (PopSize/3) est imposée pour le nombre de nouveaux descendants.

Les chromosomes de la population mère meurent dans l'ordre décroissant de la valeur de la fonction objective. C'est-à-dire que les meilleurs chromosomes sont conservés pour continuer dans la génération suivante.

La probabilité de sélection pour le croisement est donnée par (maxobj ¡ objc)/…maxobj ¡ minobj†, où minobj est la plus petite valeur de fonction objective dans la population actuelle et maxobj est la pire. Lorsque maxobj est égal à min-obj, alors la probabilité de croisement est fixée à un. Ainsi, la probabilité de sélection d'un parent dépend à la fois de sa propre valeur de fonction objectif et de la fourchette de performances de la population. Plus la valeur de la fonction objective d'un chromosome est faible, plus sa probabilité de croisement est élevée, de sorte que les «bons» gènes ont plus de chances de se recombiner à la génération suivante.

Chaque chromosome est un candidat à la mutation avec une probabilité de mutation calculée de manière similaire à la probabilité de croisement. Cependant, seule une seule paire famille/période (avec une taille de lot positive) dans chaque chromosome est mutée. Un yjt sélectionné est réduit d'un montant aléatoire limité par le niveau d'inventaire de la famille j à la période t, et yjt‡1 est augmenté du même montant, en préservant toujours la non-négativité de l'inventaire.

5. Migrations

L'opérateur de migration vise à croiser les chromosomes de populations parallèles et à ramener la progéniture dans leurs populations parentes respectives. Les chromosomes sont réparés au cas où ils violeraient les équations d'équilibre des stocks.
Les chromosomes migrés remplacent complètement les populations parentales afin d'obtenir une bonne fusion.

6. Procédure TSSA

TSSA lance la recherche à partir d'une solution initiale représentée par les tailles de lot dans un chromo sélectionné au hasard.

certains et perturbe la solution un nombre de fois donné. Chaque fois que la solution actuelle est perturbée, la taille du lot d'un article est réduite (augmentée) dans la période t1 et augmentée (réduite) dans la période t2 d'une quantité aléatoire DeltaLot où t1 et t2 sont deux périodes consécutives.

Le pseudocode pour TSSA est donné dans le tableau 3. Ici, un mouvement est considéré pour exécution s'il n'est pas trouvé parmi les mouvements de taille (Tabulist1) dans Tabulist1 qui stocke un certain nombre de couples élément/période sélectionnés (j* ; t1† dont le lot ont été déduites/augmentées d'un montant aléatoire lors de la recherche. Tabuliste1 empêche la recherche de réduire/augmenter la taille d'un lot familial récemment augmenté/réduit. La taille de Tabuliste1 est dynamique. En d'autres termes, si la solution actuelle ution ne change pas dans les N derniers déplacements, alors la taille de Tabulist1 est tronquée aléatoirement en ne conservant que les déplacements les plus récents .

Si les M derniers mouvements sont tous des mouvements de détérioration, alors size(Tabulist1) est étendu de manière aléatoire sans dépasser la taille maximale de tabuliste, MaxSize. La taille de la liste tabou dynamique est conçue pour aider la recherche à quitter les minima locaux et à l'empêcher de visiter les pires zones de la région réalisable.

Tout d'abord, il est décidé au hasard si la taille d'un lot sélectionné va être réduite ou augmentée. Ensuite, le couple …j*; t1† pas sur Tabuliste1 est sélectionné au hasard. t2 prend au hasard l'une des valeurs suivantes : t1 ¡ 1 ou t1 ‡ 1. Ensuite, un montant aléatoire, DeltaLot, qui préserve la non-négativité de l'inventaire et de la taille du lot, est sélectionné, et la modification correspondante de la valeur de la fonction objectif, CostDif, est calculé.

Une seconde tabuliste, Tabuliste2, stocke un certain nombre de valeurs de fonction objectif obtenues dans les dernières itérations de taille (Tabuliste2). Tabulist2 a une mémoire plus longue que Tabulist1 et elle est conservée pour empêcher les mouvements répétitifs ainsi que pour empêcher la recherche de considérer des solutions redondantes qui aboutissent à la même valeur de fonction objectif. La longueur de Tabulist2 est également dynamique, mais MaxSize pour Tabulist2 est supérieure à celle de Tabulist1. Tabuliste1 a une mémoire à court terme alors que Tabuliste2 en a une à long terme. Un mouvement peut être exécuté si Cost ‡ CostDif n'est pas sur Tabulist2.

Lorsque CostDif est positif, une probabilité d'acceptation, PA, est calculée pour décider si le mouvement va être exécuté. Contrairement à une procédure TS, plutôt que d'accepter le meilleur mouvement non améliorant dans tout le voisinage, la procédure TSSA calcule PA, qui dépend de l'ampleur de la détérioration causée par le mouvement dans la fonction objectif et du nombre de fois qu'un coût détérioré s'est produit. été obtenu consécutivement (tSA).

PA est calculé comme suit :

$$PA\ldots Co\hat{u}tDif\ ;\ tSA\dagger\ \hat{}\ \exp\ \ldots\text{¡}Co\hat{u}tDif = \ldots Co\hat{u}t{*}tSA\dagger\dagger$$

où Coût est le coût de la solution actuelle. Après chaque mouvement non amélioré, la température, tSA, est réduite comme suit

$$tSA\ tSA = \ldots 1 \ddagger tSA\dagger$$

où est un paramètre inférieur à un. Initialement , tSA est égal à un. Si tSA tombe en dessous d'une très petite valeur, alors il est réinitialisé à un, de sorte que PA ne devienne pas infiniment petit après un grand nombre de mouvements.

Le paramètre de réduction de température n'est pas constant. Au contraire, la valeur de est dynamique et dépend de l'état de la recherche, à l'instar des tabulistes dynamiques.
Si le dernier nombre M de déplacements consécutifs a entraîné une amélioration des valeurs de coût, alors est diminué d'un pas inférieur à 1 (un pas pratique est de 0,01). D'autre part, si le dernier nombre M de mouvements consécutifs a entraîné une désamélioration des valeurs de coût, alors est augmenté de la taille de pas. Le schéma de réduction dynamique de la température sert à mettre à jour PA en fonction de la partie spécifique de la région réalisable vers laquelle la recherche se dirige.

Pour résumer, dans TSSA, la règle de sélection de mouvement SA est combinée avec une mémoire à court terme et cela se traduit par ce que l'on appelle le TS probabiliste. D'autres améliorations sur TS et SA sont mises en œuvre grâce à l'utilisation de tailles de liste tabou adaptatives et d'un schéma de refroidissement adaptatif.

7. Résultats de calcul

Les performances de GATA sont comparées aux solutions optimales de 90 instances de test générées par Ozdamar et Bozyel (2000). Les problèmes consistent en quatre familles à ordonnancer sur un horizon de planification de six périodes. Les caractéristiques de ces problèmes sont détaillées dans Ozdamar et Bozyel (2000).

La deuxième série de 90 problèmes consiste en 14 familles et un horizon de planification de six périodes. Les 90 premiers problèmes sont résolus de manière optimale par le package d'optimisation HyperLindo, mais le deuxième ensemble de problèmes ne peut pas être résolu de manière optimale. Les résultats sont donc comparés à la meilleure solution trouvée parmi toutes les méthodes de résolution présentées ici.

O'zdamar et Bozyel (2000) ont développé pour la CLSPOS les approches suivantes : la classique Approche Hierarchical Production Planning et ses extensions, une approche itérative (IAOP) résolvant le LP relaxé du modèle P donné dans le tableau 1 où les temps de configuration sont omis, une approche d'algorithme génétique avec un codage indirect (IGA) où une solution est représentée par des priorités pour planifier les articles dans chaque

Tableau 3. Pseudocode pour TSSA.

```
Procédure TSSA ;

fIDt : capacité inutilisée restante à la période t ;
UbLot : borne supérieure de la taille du lot à transférer ;
DeltaLot : taille du lot transféré ;
Coût : coût de la solution actuelle ;
CostDif : différence de coût causée par le déménagement ;
BestCost : coût de la meilleure solution obtenue jusqu'à présent ; ;
   t2 : indice de temps ; j* : indice de famille t1
Accepter : un résultat positif du test aléatoire pour un coup qui ne s'améliore pas}

commencer {TSSA}
   si Diminuer Lot alors

      commencer à mettre à jour Tabuliste1 ; j* sélectionné* sur Tabulist1 ; Sélectionnez t1 ; j* tel que yj ;t1 Mettre
         aléatoirement t2 ; (t2 ^ t1
         ¡ 1 ou t1 ‡ 1) g {Préservation de la faisabilité de
         UbLot ^ yj ;t1 ; Sélectionnez une quantité aléatoire l'inventaire} si t1 < t2 alors UbLot ^ min fyj ;t1 ;lj ;t1 sinon
         pour DeltaLot telle que 0
         < DeltaLot µ UbLot ; Calculer CostDif par rapport à IDt1 ;IDt2 et lj t ; Si (((CostDif 0) et (Accept)) ou
         (CostDif < 0)) et (Cost ‡ CostDif not on Tabulist2) alors *
                                  ≥

            commencer la mise à jour
                  Tabulist2 ; Mettez à jour la taille (Tabulist1) et la taille (Tabulist2) si nécessaire ;
                  Mettre à jour le paramètre SA si nécessaire ;
                  Transférer DeltaLot* de la période t1 à la période t2 ; fyj ;t1 ‡
                  DeltaLot ; Ajuster IDt1 ;IDt2 ;lj ;tg ‡ DeltaLot ; yj ;t2 ^ yj ;t1 ^ yj ;t2 Coût ^ Coût ‡ CodtDif ; Si Cost *
                  yjt ; fin; fin; fin; {Diminuer Lot}
                  else {Augmenter Lot} begin ; t2 ; j* st …yj ;t2 Sélectionnez t1 > 0) et ((j*; t1) pas
            sur
         Tabulist1); Mettre à jour
   Tabuliste1 ; sélectionner

      aléatoirement t2 ; (t2 ^ t1 ¡ 1 ou t1 ‡ 1) g {Préservation de la faisabilité de l'inventaire } si t1 > t2 alors UbLot ^ min
         fyj ;t2 ;lj ;t2 sinon UbLot ^
         yj ;t2 ; Sélectionnez une quantité aléatoire pour DeltaLot
         rapport à IDt1 ;IDt2 et lj t ; Si (((CostDif 0) et (Accept)) * telle que 0 < DeltaLot µ UbLot ; Calculer CostDif par
         ou (CostDif < 0)) et (Cost
         ‡ CostDif pas sur Tabulist2) alors
                                                   *
                                  ≥
            commencer
                  Mettre à jour Tabuliste2 ;
                  Mettez à jour la taille (Tabulist1) et la taille (Tabulist2) si nécessaire ;
                  Mettre à jour le paramètre SA si nécessaire ;
                  Transférer DeltaLot de la période t2 à la période t1 ; fyj ;t1
                     *     *  yj ;t2 ^ yj ;t1 ^ yj ;t2*     *  ¡ DeltaLot ; Ajuster IDt1 ;IDt2 ;lj ;tg ‡ DeltaLot ;
                  Coût ^ Coût ‡ CoûtDif ;
                  Si Cost < BestCost alors commencez BestCost ^ Cost; Enregistrer yjkt ; fin; fin;
            fin;
         {Augmenter Lot} fin ; {TSSA}
```

(la faisabilité de l'inventaire et de la capacité est toujours préservée par l'algorithme constructif qui évalue le chromosome) et une approche de recuit simulé (PSA) pur similaire à la TSSA. Parmi ces approches, PSA, qui préserve la capacité et la faisabilité des stocks lors de l'exécution des déplacements, est signalée comme la méthode la plus performante en ce qui concerne la qualité des solutions et l'eцcacité de calcul. PSA ne tient pas de listes taboues, il part d'une solution réalisable de capacité et d'inventaire et en calculant les limites supérieures pour DeltaLot, il considère

la capacité inutilisée restante dans une période où la taille d'un lot est augmentée. Une valeur fixe de 0,01 est attribuée au paramètre dans PSA.

Les résultats de calcul donnés dans le tableau 4 consistent en les résultats obtenus par les trois méthodes, IAOP, PSA et IGA, rapportés comme les meilleurs par Ozdamar et Bozyel (2000) ainsi que les résultats obtenus par GATA. L'écart moyen en pourcentage des solutions par rapport à l'optimum et leurs écarts-types sont indiqués.

L'algorithme simple IAOP fonctionne aussi bien que l'IGA qui a été exécuté avec une taille de population de 80 et un nombre de générations maximum de 1500 pour les petits problèmes. Des résultats nettement meilleurs sont obtenus avec PSA qui a été réexécuté 15 fois chacun exécutant 10 000 mouvements. PSA commence à partir de la solution IAOP initiale.

PSA est assez eµcace en termes de temps de calcul et est remarquablement rapide. Les temps CPU fournis dans le tableau 4 sont observés sur un PC Pentium 200 Pro avec 32 Mo de RAM. En raison de sa lourde charge de calcul, les performances d'IGA seraient considérablement médiocres dans le deuxième ensemble de problèmes composé de 14 familles et de six périodes (tableau 5).

Pour comparer équitablement la procédure TSSA avec PSA, la procédure PSA est réexécutée en commençant par une solution initiale aléatoire (désignée par PSA-I dans le tableau 4). Or, ces résultats peuvent être assez comparés à la procédure TSSA*, qui fonctionne exactement de la même manière que le TSSA, mais qui préserve en permanence la faisabilité de la capacité. Bien qu'il parte d'une solution initiale aléatoire, ses performances sont nettement meilleures que celles de PSA. Ce dernier résultat est dû au paramètre SA dynamique, , qui est mis à jour après cinq ine¶efficaces (ne changeant pas la solution actuelle ou détérioration) et les deux listes tabou dynamiques prenant en charge la TSSA. Les tailles de tabuliste minimales (- maximales) pour Tabulist1 et Tabulist2 sont respectivement 5(10) et 10(100).

Pour démontrer les e¶ets sur les performances, GATA est exécuté avec di¶érents paramètres. GATA-I a une seule population (aucune migration ne se produit), mais afin de comparer avec GATA-III qui a trois populations migrantes, GATA-I est exécuté trois fois, chaque fois avec une population initiale aléatoire di¶érente. Il en va de même pour GATA-II où une seule population existe, mais en plus, GATA-II utilise 10 chromosomes fournis par la procédure TSSA*. GATA-III applique la migration toutes les 25 générations (GenNo ˆ 25). Toutes les exécutions GATA sont effectuées avec r ˆ 0:1, c'est-à-dire que 10 % de la population subit une procédure TSSA chaque fois que la TSSA est activée. Dans toutes les variantes GATA, la mutation est appliquée lorsque PopRange est supérieur à 0,7 et TSSA est appliqué lorsque PopRange est supérieur à 0,5. Tous ces paramètres sont définis en testant GATA au préalable sur des problèmes plus difficiles dans l'ensemble de test où toutes les méthodes ont trouvé des diµcultés à converger vers la solution optimale.

Une observation qui n'est pas indiquée dans le tableau 4 est que GATA-I, GATA-II et GATA-III trouvent leurs meilleures solutions dans 25, 22 et 12 générations en moyenne, respectivement. Ainsi, GATA converge assez rapidement. GATA-III se termine lorsqu'une meilleure solution n'est pas obtenue lors de la prochaine migration tandis que GATA-I et GATA-II se terminent lorsqu'une meilleure solution n'est pas obtenue dans trois générations consécutives. Le temps de calcul requis pour GATA est également modeste, car TSSA est activé une fois toutes les 10 générations en moyenne et ses mouveme

Tableau 4. Résultats pour les 90 problèmes tests (avec quatre familles, six périodes) générés par Ozdamar et Bozyel (2000).

Résultats pour 90 problèmes (4 it.x6per.)	Résultats précédents [O« zdamar, Bozyel (1996)]			Nouveaux résultats				
Procédure de solution	IAOP	IGA	Message d'intérêt public	PSA-I	TSSA*	GATA I	GATA II	GATA III
% d'écart par rapport à l'optimum (écart-type)	5,46 (5,43)	5,62 (10,08)	1,08 (4,02)	4,37 (6,34)	0,20 (0,38)	0,15 (0,30)	0,04 (0,04)	0.0
Nombre de solutions optimales	15	25	39	34	86	84	89	90
Nombre d'itérations	3 à 4 solutions PL	1500	15 ¤104	15 ¤104	15 ¤104	3 fois rediffusion	3 fois rediffusion	1 course
Popsize/PopNo	-	80/1	-	-	-	50/1	50/1	50/3
Propriété de la solution initiale	LP	aléatoire	IAOP	aléatoire	aléatoire	aléatoire	aléatoire 1 10 solutions TSSA*	aléatoire
Moyenne CPUsecs/ problème	2	420	1	1	1.3	1.8	2.6	5.4

Tableau 5. Résultats pour les 90 problèmes tests (avec 14 familles, 6 périodes) générés par Ozdamar et Bozyel (2000).

Procédure de solution	IAOP	IGA	[illegible]	TSSA*	GATA I	GATA II	GATA III
% d'écart par rapport à l'optimum (écart type)	3,59 (2,59)	78,70 (77,73)	1.33 (1.73)	2,48 (2,68)	0,88 (1,05)	0,50 (0,86)	0,17 (0,36)
Nombre d'itérations	Solution 3-4 PL	5000	150 000	150 000	3 fois rediffusion	3 fois rediffusion	1 course
PopTaille/PopNon	-	40/1	-	-	150/1	150/1	150/3
Propriété de la solution initiale	LP	aléatoire	IAOP aléatoire		aléatoire	aléatoire 1 10 solutions TSSA*	aléatoire
Moyenne CPUsecs/ problème	3	1176	1.25	3.2	4.8	6.7	8.3

limité à 20 000. Par conséquent, GATA-III exécute TSSA sur 5 chromosomes en moyenne jusqu'à ce que la meilleure solution soit obtenue. GATA-III est exécuté sur des processeurs parallèles émulés sur un réseau local de PC.

Ainsi, un problème de test est résolu en quelques secondes. L'écart moyen en pourcentage par rapport à l'optimum montre également que GATA, qui est un GA hybride intégrant les fonctionnalités TS et SA, surpasse toutes les méthodes développées précédemment.

Dans le tableau 5, les résultats obtenus sur la deuxième série de 90 problèmes sont donnés. Ici, les performances sont évaluées par rapport à la meilleure solution obtenue par toutes les méthodes présentées ici. Dans cette expérimentation, 5% de la population subit une procédure TSSA qui exécute 40 000 mouvements. Lorsque des tests t sont appliqués pour vérifier les différences statistiquement significatives entre les moyennes de chaque paire de méthodes, nous obtenons les résultats suivants pour le premier ensemble de 90 problèmes. Des di¶érences statistiquement significatives existent entre les moyennes de toutes les paires de méthodes à tous les niveaux de signification sauf TSSA/GATA-I (insignifiant), PSA/GATA-I (niveau de signification de 5 %) et TSSA/PSA (2,5 % niveau de signification).

Dans la deuxième série de 90 problèmes, à l'exception de PSA/GATA-I qui est significatif à un niveau de signification de 2,5 %, toutes les paires de méthodes ont des di¶érences moyennes statistiquement significatives à tous les niveaux de signification. Ces résultats montrent que dans les petits problèmes, GATA-I n'est pas efficace par rapport à TSSA*. De plus, le classement dans la performance de TSSA* et PSA est permuté lorsque des problèmes de plus grande taille sont considérés.

Les listes tabou semblent limiter les capacités de recherche de SA dans les problèmes plus importants. En observant les résultats des deux problèmes de test, on peut conclure qu'il vaut certainement la peine d'utiliser les procédures GATA-II et GATA-III malgré l'e¶ort de calcul supplémentaire.

8.Conclusion

Une procédure GA (GATA) est développée intégrée avec Tabu Search and Simulated Annealing (TSSA) pour résoudre un problème de planification de la production qui est assez courant dans la pratique. GATA trouve des résultats optimaux pour des problèmes de test sur lesquels d'autres méthodes ont été largement testées dans une étude précédente (Ozdamar et Bozyel 2000). Dans cette dernière référence, l'expérience informatique démontre qu'un GA qui utilise un codage indirect est incapable de faire face au CLSPOS. Le codage direct de GATA préserve les bons schémas qui sont implicitement à l'intérieur des chromosomes. Cependant, comme il est diµcile de construire des solutions réalisables en codage direct, TSSA permet de visiter le plus possible des solutions réalisables au voisinage des solutions infaisables correspondant à la plupart des chromosomes de la population. Les résultats encourageants obtenus par GATA suggèrent que les heuristiques de recherche hybrides qui intègrent GA, TS et SA peuvent réussir à traiter d'autres problèmes NP-difficiles dans la planification et le contrôle de la production.

Les références

Alexandre, F., Cardeira, C., Charpillet, F., Mammeri, Z. et Portmann, MC, 1997, Compu-Search Methodologies II : ordonnancement à l'aide d'algorithmes génétiques et de réseaux de neurones artificiels. Dans A. Artiba et SE Elmaghraby (eds). La planification et l'ordonnancement des systèmes de production : méthodologies et applications (Chapman et Hall) Chapitre 10, pp. 301–336.

Bitran, GR, et Yanasse, HH, 1982, Computational com- plexity of the capacitated lot size problem. Sciences de gestion, 28, 1174–1186.

Catrysse, D., Maes, J. et Van Wassenhove, LN, 1990, Set partitioning and column generation heuristics for capacitated lotsizing. Revue européenne de recherche opérationnelle, 46, 38–47.

Diaby, M., Bahl, HC, Karwan, MH et Zionts, S., 1992, Une approche de relaxation lagrangienne pour le dimensionnement de lots capacitifs à très grande échelle. Sciences de gestion, 38, 1329–1340.

Gu'nther, HO, 1989, Planification de la taille des lots et des besoins en capacité dans un système de production en une seule étape. Journal européen de recherche opérationnelle, 31, 223–231.

Maes, J., et Van Wassenhove, LN, 1986, A simple heur istic for the multi-item single level capacitated lot sizing problem. Lettres de recherche opérationnelle, 4, 265–273.

O'zdamar, L., et Bozyel, MA, 2000, Le problème de dimensionnement de lot capacité avec des décisions d'heures supplémentaires et des temps d'installation. Pour apparaître dans les transactions IIE.

Portmann, MC, 1996, Algorithmes génétiques et ordonnancement : un état de l'art et quelques propositions, Actes de l'atelier sur la planification et le contrôle de la production, Mons, Belgique, 9–11 septembre, pp. I–XIV.

Trigeiro, WW, Thomas, LJ et McLain, JO, 1989, Dimensionnement de lot capacité avec temps de configuration. Sciences de gestion, 35, 353–366.