Recherche de la meilleure combinaison de scénarios en simulation-optimisation

JEAN WERY[1], JONATHAN GAUDREAULT[1,2], CORINNE CHABOT[3], ANDRÉ THOMAS[4], CLAUDE-GUY QUIMPER[1,2], PHILIPPE MARIER[1]

[1] Consortium de recherche FORAC
Université Laval
1065, av. de la Médecine, Québec (QC) G1P 0A6, Canada
Jean.Wery@cirrelt.ca, Philippe.Marier@forac.ulaval.ca

[2] Département d'informatique et de génie logiciel
Université Laval
1065, av. de la Médecine, Québec (QC) G1P 0A6, Canada
Jonathan.Gaudreault@ift.ulaval.ca, Claude-Guy.Quimper@ift.ulaval.ca

[3] Centre de Recherche Industrielle du Québec
333, rue Franquet, Québec (QC) G1P 4C7, Canada
Corinne.Chabot@criq.qc.ca

[4] Centre de Recherche en Automatique de Nancy
Université de Lorraine
Faculté des sciences - BP 239, 54506 Vandoeuvre les Nancy, France
Andre.Thomas@univ-lorraine.fr

Résumé **- Lorsque le temps est manquant, la simulation-optimisation est une méthode très utilisée pour déterminer le « meilleur » scénario possible. Or, lorsque l'on recherche la meilleure combinaison de scénarios et non le meilleur scénario, les méthodes classiques ne semblent pas pouvoir s'appliquer. Dans la littérature, le problème de la recherche du meilleur ensemble de scénarios ne semble pas avoir été introduit formellement. Nous proposons une définition formelle de ce problème. De plus, nous pensons que certaines méthodes de recherche dans les arbres, issues de la programmation par contraintes pourraient être efficaces pour résoudre ce problème. Pour illustrer le problème et vérifier la pertinence de ces méthodes de résolution, la démarche est appliquée pour déterminer la meilleure séquence de configuration d'un système de découpe de bois de plancher. L'étude réalisée montre que les méthodes issues de la programmation par contraintes pourraient se révéler efficaces pour résoudre ce type de problèmes. En effet, la méthode *Limited Discrepancy Search* (LDS) obtient des résultats très semblables à une heuristique spécialement élaborée pour le cas étudié. Or LDS est une méthode générique et pourrait s'appliquer à d'autres cas.**

Abstract **- When time is missing, simulation optimization is a widely used method to determine the "best" possible scenario. However, when the best combination of scenarios is sought (instead of the best scenario), conventional methods do not seem to be applicable. In the literature, the problem of finding the best set of scenarios does not seem to have been formally introduced. We propose a formal definition of this problem. In addition, we believe that some search tree methods usually used in constraint programming could be effective in solving such problem. To illustrate the problem and to verify the relevance of these methods of resolution, the approach is applied to determine the best sequence of configurations of a wood flooring system. The study shows that methods used in constraint programming could be effective in solving this type of problem. Indeed, the *Limited Discrepancy Search* (LDS) method obtains results very similar to a heuristic specially developed for the studied case. LDS is a generic method and could be applied to other cases.**

Mots clés **- Simulation-optimisation, Programmation par contraintes, Simulation, Optimisation, Recherche dans les arbres.**

Keywords **– Simulation optimization, Constraint programming, Simulation, Optimization, Search tree.**

1 INTRODUCTION

En industrie, la simulation est une méthode majoritairement utilisée pour connaître les réactions d'un système de production à certains facteurs (ou stratégies) sans avoir à réaliser d'essais physiques (qui seraient souvent trop coûteux, voire impossibles). Lorsque le temps imparti pour la simulation ne permet pas de simuler toutes les possibilités, les méthodes dites de « simulation-optimisation » sont alors utilisées. Elles permettent de déterminer dynamiquement les prochains scénarios (représentation d'un système dans une configuration donnée dans un modèle de simulation) qui devraient être simulés, le but étant de trouver la meilleure solution possible avec le temps de simulation disponible. Les méthodes classiques de simulation-optimisation se basent sur les résultats de simulations déjà effectuées pour guider la

recherche i.e. pour déterminer quel sera le prochain scénario à évaluer. Cependant, il n'est pas toujours possible de les utiliser pour résoudre des problèmes tels que le problème de la recherche de la meilleure combinaison de scénarios. Certaines méthodes issues de la PPC (Programmation Par Contraintes) pourraient être utilisées pour chercher des solutions dans cet espace.

Lorsque l'on cherche à trouver la meilleure combinaison de scénarios tout en évaluant seulement un petit nombre de scénarios dans un temps donné et que le simulateur utilisé ne nous permet pas de simuler directement une combinaison de ces scénarios, les méthodes de simulation-optimisation classique ne peuvent être utilisées. Nous proposons donc de vérifier si l'utilisation de méthodes issues de la PPC pourrait être pertinente pour résoudre ce type de problèmes. En effet, certains algorithmes couramment utilisés en PPC nous permettent de prioriser les scénarios qui devraient être simulés. Nous pensons que leur application pour déterminer l'ordre dans lequel les scénarios devraient être simulés peut permettre de trouver une bonne solution dans le temps imparti.

Dans un premier temps, nous présentons la simulation-optimisation classique et les différentes techniques associées. Ensuite, le problème de la recherche de la meilleure combinaison de scénarios est introduit (section 3), puis nous présentons un cas d'étude basé sur un simulateur de bois de plancher pour illustrer ce problème (section 4). Ensuite, différentes méthodes issues de la PPC qui pourraient se révéler efficaces pour la résolution d'un tel problème sont décrites (section 5). L'approche retenue dans le cadre de ce papier et le protocole expérimental sont expliqués (section 6). Enfin, les résultats (section 7) sont présentés et discutés.

2 LA SIMULATION-OPTIMISATION EN CONTEXTE CLASSIQUE

La combinaison de la simulation avec l'optimisation est une approche de plus en plus utilisée. En effet, cette approche permet d'utiliser les points forts de chaque technique pour obtenir un résultat qui serait très difficile à obtenir sans cela. Ladier et al. (2014) identifient quatre différents types de relations entre la simulation et l'optimisation : (1) l'optimisation intégrée au modèle de simulation, (2) le modèle de simulation créant et transmettant des données servant à l'optimisation, (3) la simulation intégrée au modèle d'optimisation, (4) la simulation évaluant les résultats de l'optimisation. La simulation permet donc d'obtenir des informations qui peuvent ensuite être utilisées dans les modèles d'optimisation comme intrants.

Traditionnellement, lorsque le temps est limité ou lorsque le budget n'est pas suffisant pour simuler tous les scénarios, on utilise des méthodes de *simulation-optimisation*. Dans la littérature, on retrouve différentes variantes de cette approche. Carson et Maria (1997) définissent la simulation-optimisation comme étant le processus d'affectation des meilleures valeurs possibles à des variables d'entrées sans avoir à évaluer explicitement toutes les combinaisons possibles. On cherche bien sûr à trouver la meilleure solution compte tenu du temps disponible. April et al. (2003) la définissent plutôt comme le moyen de trouver le modèle de simulation, parmi les différents modèles possibles, qui permettra d'obtenir des performances optimales. Jian et Henderson (2015) indiquent que si des décisions peuvent être représentées comme des variables de décisions dans un modèle de simulation, alors il est possible de faire de la simulation-optimisation, i.e. de choisir les variables de décisions dans le but de maximiser ou minimiser des mesures de performances estimées grâce à la simulation. Fu (2015), dans l'introduction de son livre *Handbook of simulation optimization* semble avoir la même vision que ces derniers tout en faisant remarquer que le terme est parfois utilisé pour parler du processus de recherche en lui-même.

Dans les faits, la simulation-optimisation désigne tout simplement la résolution d'un problème d'optimisation dans un contexte où l'évaluation de la fonction-objectif nécessite l'exécution d'un modèle de simulation. Or, puisque le nombre total de simulations à réaliser est trop élevé pour se permettre d'évaluer toutes les alternatives possibles, une procédure de recherche (globale ou locale) doit être utilisée pour décider quels scénarios seront évalués et dans quel ordre (Figure 1).

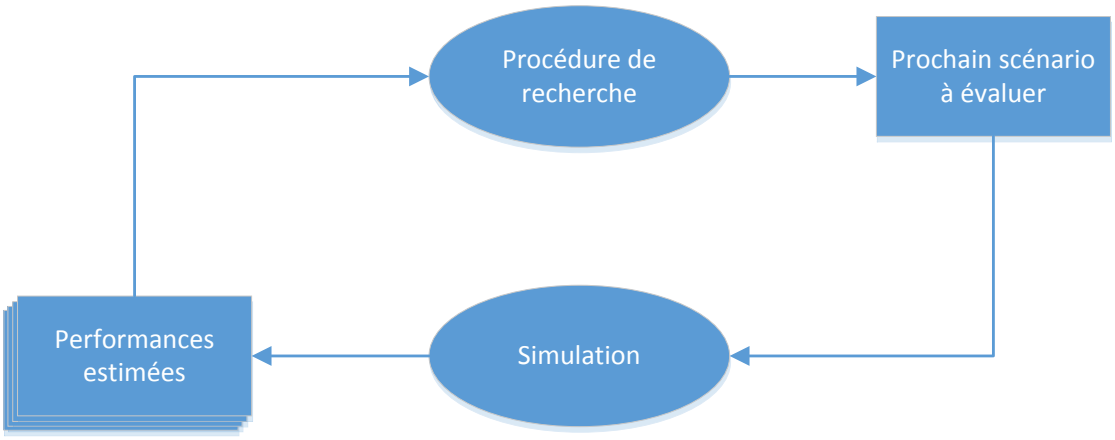

Figure 1. Principe de fonctionnement des méthodes de simulation-optimisation, adapté de Fu (2002)

2.1 *Forme générale d'un problème de simulation-optimisation classique*

La formulation générale d'un problème de simulation-optimisation telle qu'on la trouve dans la littérature est la suivante. Il s'agit d'un problème d'optimisation dans lequel on cherche à minimiser ou maximiser la valeur d'une fonction objectif *f*, où *x* est un vecteur représentant les variables de décisions et *X* l'espace des solutions [Jian et Henderson, 2015, Fu, 2015, Better et al., 2008] :

$$\underset{x \in X}{Max}\ f(x)$$

Cependant, la fonction *f* n'est pas directement évaluable (car elle implique un phénomène aléatoire) mais sa valeur peut être estimée grâce à la simulation. La fonction *f* peut donc être représentée comme l'espérance d'un résultat de simulation (réponse) pour une variable aléatoire de sortie $Y(x,\xi)$, où ξ est l'ensemble des nombres aléatoires d'une simulation :

$$f(x) = E\left[Y(x,\ \xi)\right]$$

En pratique, nous disposons d'une fonction de simulation $S(x,\xi_i)$ qui retourne, pour un scénario *x* et une réalisation ξ_i des paramètres aléatoires, un vecteur de métriques servant à évaluer cette réalisation du scénario. $F(S(x,\xi_i))$ prend en entrée ces résultats et en évalue la valeur selon le critère établi par le décideur. Chaque scénario sera évalué pour *n* réalisations des paramètres aléatoires. Chacune de ces simulations constituera une *réplication*. Nous supposerons donc que :

$$f(x) \approx F\left(S\left(x,\xi\right)\right) = \frac{1}{n}\sum_{i=1}^{n} F\left(S(x,\xi_i)\right)$$

Et nous chercherons à maximiser cette valeur :

$$\underset{x \in X}{Max}\ F\left(S\left(x,\xi\right)\right)$$

Par exemple, supposons le problème du design d'une usine. Nous cherchons le design qui va maximiser la performance *F* de l'usine (Figure 2). Pour chaque configuration du système possible (i.e. chaque design différent, chaque combinaison de différents paramètres sur lesquels on peut effectuer des changements d'états), nous définirons un scénario de simulation *x* d'usine, $x \in X$. Pour chaque scénario *x*, nous pouvons simuler le comportement du système (i.e. l'usine) plusieurs fois (avec des nombres aléatoires ξ différents à chaque réplication). $S(x_i,\xi)$ est le résultat de la simulation d'un scénario x_i. Dans ceci, les différentes valeurs de ξ pourraient correspondre à des taux de pannes de machines ou encore à des délais de livraison. Certains procédés ou même la demande ne sont pas constants et ont une part aléatoire; on réalisera alors un nombre *n* de réplications dans le but d'estimer la valeur de la réponse.

2.2 *Techniques utilisées en simulation-optimisation classique*

Lorsque le temps disponible ne permet pas de simuler toutes les réplications nécessaires (pour chaque scénario, souvent appelé « alternative ») pour déterminer la « meilleure » alternative (i.e. pour trouver le « meilleur » scénario) mais qu'il est tout de même possible de réaliser un certain nombre de réplications pour estimer la performance de chaque scénario, en simulation-optimisation, les techniques de *classification et sélection* [Goldsman et Nelson, 1994, Law, 2007] sont très utilisées. Elles permettent de déterminer dynamiquement le ou les scénario(s) pour lesquels on doit augmenter le nombre de réplications. En effet, lorsque la simulation est utilisée pour comparer différentes alternatives, il est courant d'augmenter le nombre de réplications pour chaque scénario pour obtenir une variance de l'espérance d'un résultat de simulation (réponse) suffisamment petite pour pouvoir déterminer le meilleur scénario. L'approche la plus évidente est de choisir un nombre de réplications identiques pour chaque scénario. Or, cette technique peut être relativement inefficace. En effet, si un scénario a une faible variance, sa performance peut être estimée grâce à un petit nombre de réplications. C'est le principe des techniques de classification et sélection. Deux approches sont principalement utilisées, *Optimal Computing Budget Allocation* (OCBA) et *Algorithm and Expected Value of Information* (EVI) [Chen et al., 2015].
Il existe également des techniques destinées aux situations dans lesquelles le budget de simulation ne permet même pas de tester chaque scénario au moins une fois. Fu (2015) traite de ces différentes techniques. Les quatre grandes méthodes qui y sont traitées sont la méthode de la surface de réponse, l'approximation stochastique, l'approximation moyenne de l'échantillon et les méthodes de recherche aléatoire.

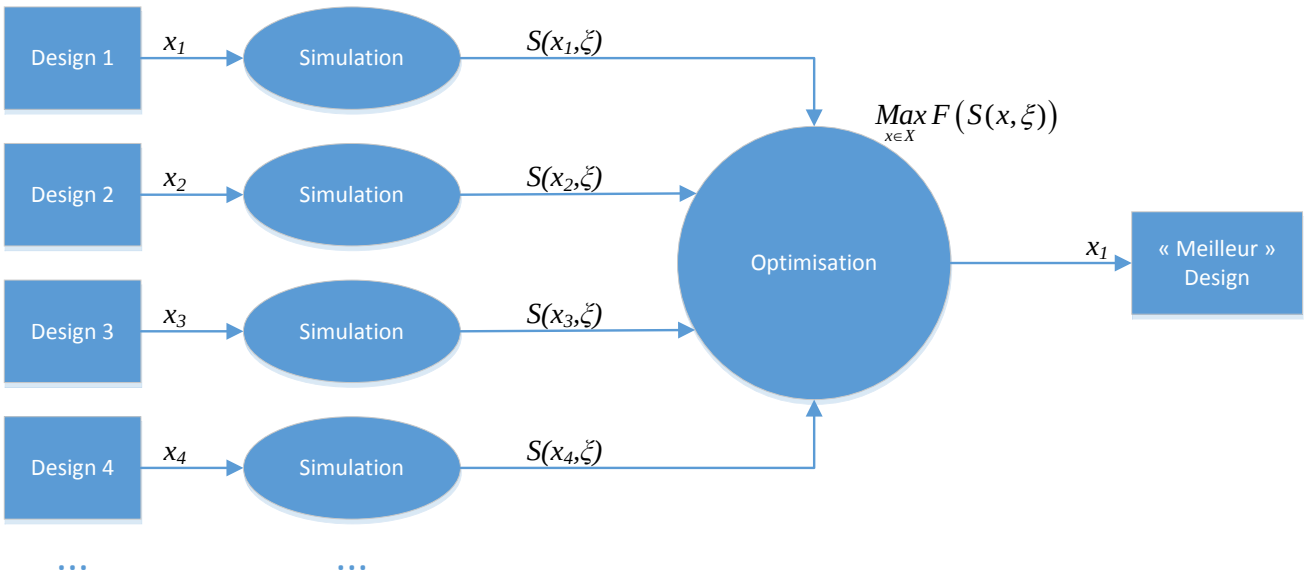

Figure 2. Méthode pour évaluer et déterminer le « meilleur » design d'une usine

La méthode de la surface de réponse (en anglais, *Response Surface Methodology*, RSM) utilise le modèle de simulation comme une boîte noire. Les entrées et sorties du modèle de simulation sont les seules informations qui sont exploitées [Kleijnen, 2015]. Elle se base sur les résultats d'un plan d'expériences pour créer un métamodèle. Un métamodèle est une relation mathématique qui approxime la relation entre l'entrée (i.e. scénario ou combinaisons d'intrants du modèle de simulation) et la sortie (espérance du résultat de la simulation), c'est-à-dire un modèle simplifié du modèle originel de simulation [Kleijnen et al., 2005]. Dans le cas de la méthode de la surface de réponse, les métamodèles sont des modèles de régression linéaire élaborés à partir des résultats d'un plan d'expériences qui définit les scénarios à simuler [Kelton et Barton, 2003]. La méthode de la surface de réponse est considérée par Kleijnen (2015) comme une heuristique séquentielle, car elle utilise une séquence d'expérimentations locales pour chercher à déterminer la combinaison optimale des intrants de la simulation. Elle va donc permettre d'estimer la combinaison d'intrants (le scénario) qui minimise (ou maximise) la fonction objectif. Cependant, cette méthode ne garantit pas de trouver l'optimalité [Kleijnen, 2015].

L'utilisation d'un métamodèle permet de simplifier la résolution d'un problème de simulation-optimisation. En effet, plutôt que d'être stochastique, la réponse du métamodèle est déterministe, et son utilisation est beaucoup plus rapide que l'exécution du modèle de simulation initial [Barton et Meckesheimer, 2006].

L'approximation stochastique (*Stochastic Approximation*, SA) est un algorithme de recherche itératif basé sur le gradient. En utilisant des méthodes statistiques rigoureuses, l'algorithme va en quelque sorte imiter l'algorithme du gradient (utilisé classiquement en optimisation déterministe) pour prendre en compte les composantes stochastiques du problème [Fu, 2002]. Deux méthodes classiques sont principalement utilisées : *Robbins–Monro* (RM) [Robbins et Monro, 1951] et *Kiefer–Wolfowitz* (KW) [Kiefer et Wolfowitz, 1952].

L'approximation moyenne de l'échantillon (en anglais *Sample Average Approximation*, SAA, parfois trouvé sous le nom de *Sample Path Optimization*) est aussi basée sur le gradient. Cependant, ce n'est pas vraiment un algorithme. C'est plutôt une « recette » à utiliser pour faire face à des problèmes de simulation-optimisation. Cette technique se base principalement sur le fait que la fonction $f(x)$ que l'on cherche à estimer par simulation peut être approximée en faisant la moyenne des résultats de toutes les réplications pour chaque scénario [Jian et Henderson, 2015]. Cette estimation s'explique par la loi des grands nombres. En effet, si l'on répète un nombre de fois suffisamment élevé une expérience (i.e. beaucoup de réplications d'un même scénario avec des nombres aléatoires différents), on devrait se rapprocher de la moyenne. De ce fait, l'optimisation qui en découle peut être réalisée en se basant sur la moyenne trouvée [Shapiro, 2013]. Il est cependant très important d'utiliser des CRN (*Common Random Numbers*) i.e. des nombres aléatoires « identiques » pour chaque scénario différent [Jian et Henderson, 2015]. Il faut aussi veiller à bien identifier la taille de l'échantillon (i.e. que le nombre de réplications à effectuer soit suffisamment grand), pour être sûr que la solution trouvée soit bien une solution optimale au problème initial [Shapiro, 2013]. Cependant, l'application de l'approximation moyenne de l'échantillon montre que cette technique n'est pas aussi efficace que l'approximation stochastique, mais que de récentes extensions à la méthode de base peuvent permettre de rivaliser avec celle-ci [Kim et al., 2015].

Les méthodes de recherche aléatoire ont été développées pour résoudre des problèmes d'optimisation déterministes. Elles ont été adaptées pour résoudre des problèmes de simulation-optimisation pour lesquels la fonction objectif ne peut pas être évaluée de façon déterministe [Andradóttir, 2006, Fu et al., 2005]. Les méthodes de recherche aléatoire peuvent être utilisées pour résoudre la plupart des problèmes de simulation-optimisation (que le nombre de combinaisons d'intrants du modèle de simulation soit fini ou pseudo-infini i.e. non comptable). Elles sont bien adaptées pour résoudre des problèmes dont la fonction objectif a une structure peu connue et dont la procédure d'optimisation doit identifier de meilleures solutions tout en étant guidée par la performance estimée des solutions déjà considérées [Andradóttir, 2015]. Elles convergent généralement vers des solutions optimales et peuvent mener à des optimums locaux ou globaux suivant la méthode utilisée [Andradóttir, 2006].

Les méthodes de recherche aléatoire permettent de se déplacer d'une solution candidate à une autre dans l'espace de recherche i.e. lorsqu'une combinaison d'intrants a été évaluée par simulation, ces méthodes vont permettre de trouver une autre combinaison d'intrants (possiblement meilleure) qui devrait être évaluée. Le déplacement vers l'autre solution candidate va généralement se faire à l'aide de techniques statistiques [Fu et al., 2005].

Selon Fu et al. (2005), l'algorithme de recherche doit avoir deux caractéristiques principales: il doit définir comment la prochaine solution candidate est choisie et comment déterminer quelle est la meilleure solution actuelle. Le *branch-and-bound*, la recherche tabu et les algorithmes génétiques sont des exemples de méthodes de recherche aléatoire.

3 LE PROBLEME DE RECHERCHE DE LA MEILLEURE COMBINAISON DE SCENARIOS

En simulation-optimisation classique, on cherche le ou les scénarios maximisant une fonction-objectif. Dans le cadre de cet article, nous recherchons plutôt la « combinaison » de scénarios qui maximisent *conjointement* une fonction-objectif. Reprenons le cas de design d'usines et imaginons maintenant que nous devons construire plusieurs usines qui permettront conjointement de générer les meilleurs profits pour la compagnie (plutôt que de chercher la meilleure configuration d'un système, nous cherchons les *m* configurations de ce système qui, ensemble, donnent le meilleur résultat). Dans ce cas précis, il serait toujours possible de simuler directement chacune des combinaisons possibles. Chaque ensemble qui serait un assemblage de sous-modèles de simulation correspond alors à une alternative. Le nombre de combinaisons possibles étant très élevé, le nombre de scénarios à simuler est très grand et la modélisation de chaque alternative peut être complexe. Dans certains cas, il n'est pas envisageable de le faire, car cela serait trop consommateur en temps de modélisation ou bien le simulateur utilisé n'est tout simplement pas adapté pour le faire.

Ce problème ne semble pas avoir été considéré dans la littérature. C'est pourquoi nous proposons un cadre formel pour l'adresser. Nous proposons de représenter ce problème de la manière suivante :

$$\underset{\bar{X}\subseteq X \,||\bar{X}|\leq\theta}{Max}\; g\left(\bar{X}\right)$$

$$\text{Avec } g\left(\bar{X}\right)\approx G\left(\left\{S(x,\xi)\right\} \mid x\in\bar{X}\right)$$

Où $S(x,\xi)$ est le vecteur de taille n des résultats de simulation $S(x,\xi_i)$, $\bar{X}$ est l'ensemble des scénarios recherchés et doit être de taille maximale Θ, $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ est la fonction permettant d'évaluer un ensemble de scénarios.
La figure 3 illustre cette démarche dans un cas de design d'usines dans un réseau.
Les méthodes utilisées en simulation-optimisation ne nous semblent pas adaptées pour faire face à ces problèmes combinatoires. En effet, les méthodes de simulation-optimisation classique se basent généralement sur l'évaluation du résultat, le « score », d'une simulation (exemple : la performance d'une usine). Or, dans le cas présenté, ce sont des ensembles de scénarios qui sont évalués et comparés par l'entremise de l'optimisation pour trouver le meilleur d'entre eux. Nous ne sommes donc pas en mesure d'obtenir directement le score associé à une simulation. Or, en simulation-optimisation classique, à partir de ces scores, les scénarios de simulation sont analysés pour déterminer la prochaine alternative à évaluer. Cela semble difficilement applicable pour la recherche de la meilleure combinaison de scénarios.

4 Presentation du cas d'etude

La plupart des usines de bois de plancher utilisent des systèmes permettant de découper les planches de bois fournies en entrée en produits de plus petites tailles afin de maximiser la valeur produite. Ces systèmes scannent chaque planche et prennent une décision concernant la découpe une planche à la fois.
Avant de lancer la production, on doit fournir à celui-ci la liste x des produits qu'il lui est permis de fabriquer. Cette liste de produits dits « actifs » constitue la configuration du système. On peut fournir au système une « configuration » qui sera active pour la totalité du quart de travail, ou encore fragmenter le quart de travail en plusieurs périodes et utiliser une configuration différente à chaque période. Le problème $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ que l'on cherche à résoudre est justement de déterminer quelles configurations (i.e. liste de produits actifs) devraient être utilisées pour chaque période de manière à maximiser la valeur totale de la production tout en respectant des contraintes de marché (ex : pas plus de Z unités du produit W doivent être fabriqués, autant de produits U que de V doivent être réalisés, etc.). Or, nous ne savons pas, a priori, quelles sont les quantités de chaque produit que l'on peut espérer fabriquer lorsqu'une configuration donnée x_i est active. On doit donc utiliser la simulation $S(x,\xi)$ pour l'anticiper, mais il est impossible de simuler au préalable la totalité des configurations possibles.
En effet, il est possible pour une entreprise d'avoir plus de 100 produits différents ce qui implique d'avoir 2^{100} combinaisons de produits actifs/inactifs différentes (2^{100} configurations, donc). Il faudrait environ dix mille milliards de milliards d'années pour simuler toutes les combinaisons possibles sur un ordinateur standard.
Il est donc nécessaire d'imaginer une procédure pour sélectionner les combinaisons à simuler. Or, contrairement aux problèmes résolus par la simulation-optimisation classique, la valeur de $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ ne dépend pas de x_i seul, mais bien de tous les x_i testés jusqu'à maintenant puisque $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ est un problème d'optimisation combinatoire. Il est donc apparemment très difficile d'imaginer comment se baser sur les résultats de $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ pour déterminer les prochains x_i à simuler.
Une alternative serait de tester au hasard (compte tenu du temps de simulation alloué) un certain nombre (θ) de configurations. Cependant, on peut imaginer que certaines configurations sont susceptibles d'être plus utiles que d'autres. Par exemple, la configuration où tous les produits sont inactifs risque d'être fort inutile. Dans la section suivante, nous montrons qu'il est possible de représenter formellement l'espace des configurations possibles sous la forme d'un arbre pour lequel chaque feuille constitue une configuration possible. Nous évaluerons par la suite différents algorithmes d'exploration de cet arbre quant à leur propension à énumérer les feuilles en ordre décroissant d'utilité.

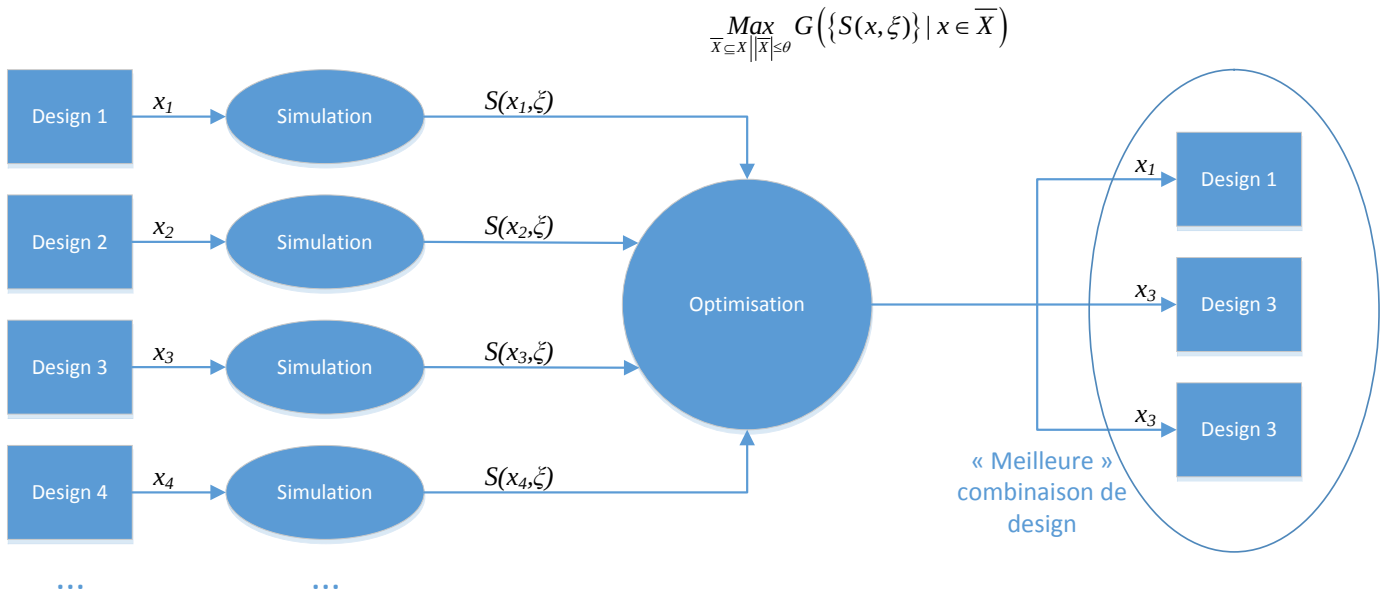

Figure 3. Recherche de la meilleure combinaison de scénarios pour le design de plusieurs usines dans un réseau

5 REPRESENTER L'ESPACE DES CONFIGURATIONS POSSIBLES SOUS LA FORME D'UN ARBRE

En programmation par contraintes, les problèmes peuvent être modélisés sous la forme d'un arbre. Dans notre cas, l'ensemble complet des configurations peut être représenté sous la forme d'un arbre où chaque nœud correspond à un produit, chaque arc correspond à l'activation ou la désactivation de ce produit, et chaque feuille de l'arbre correspond donc à un scénario qui pourrait être simulé (Figure 4).

En programmation par contraintes, des méthodes ont été développées pour fouiller dans cet arbre. Le parcours en profondeur (*Depth First search*, ou DFS) est la méthode de base permettant d'énumérer toutes les configurations possibles. Si le temps de simulation est limité, nous visiterons donc les *n* configurations les plus à gauche. Or, de cette manière, les scénarios simulés risquent d'être très semblables les uns par rapport aux autres.

Or, il existe des techniques d'exploration d'un arbre permettant, si le temps est limité, d'explorer un échantillon beaucoup plus représentatif de l'ensemble des feuilles. C'est le cas de l'algorithme LDS (*Limited Discrepancy Search*). Introduit par Harvey et Ginsberg (1995) puis amélioré par Korf (1996), LDS est une stratégie de recherche dans les arbres qui permet de visiter en un temps donné un ensemble de feuilles beaucoup plus hétérogène que ne le ferait DFS (Figure 5). LDS est un algorithme itératif. On visite d'abord la feuille la plus à gauche (dans notre exemple, celle pour laquelle tous les produits sont actifs). Puis, on exécute un DFS, mais en permettant seulement d'atteindre les feuilles pour lesquelles il y a un seul branchement à droite sur le chemin allant de la racine jusqu'à la feuille. Lors de cette itération, nous visitons donc les feuilles pour lesquelles un seul produit est inactif. On lance donc à nouveau DFS afin de visiter les feuilles avec 2 produits inactifs, etc.

D'autres variantes de cette approche (ex : DDS, DBDFS, IDFS) ont ensuite été proposées par différents auteurs. Elles sont recensées par Barták (2004).

Par ailleurs, il est à noter que changer l'ordre dans lequel les produits sont énumérés (i.e. quel est le premier produit placé au haut de l'arbre, le deuxième, etc.) change la topologie de l'arbre et donc l'ordre dans lequel les solutions seront énumérées.

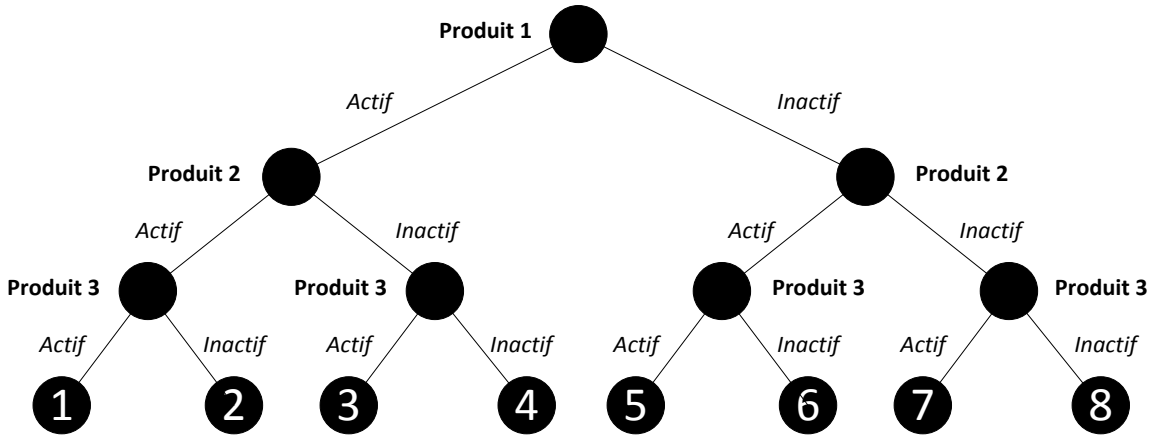

Figure 4. Arbre de recherche pour le problème de configuration d'usine de bois de plancher.

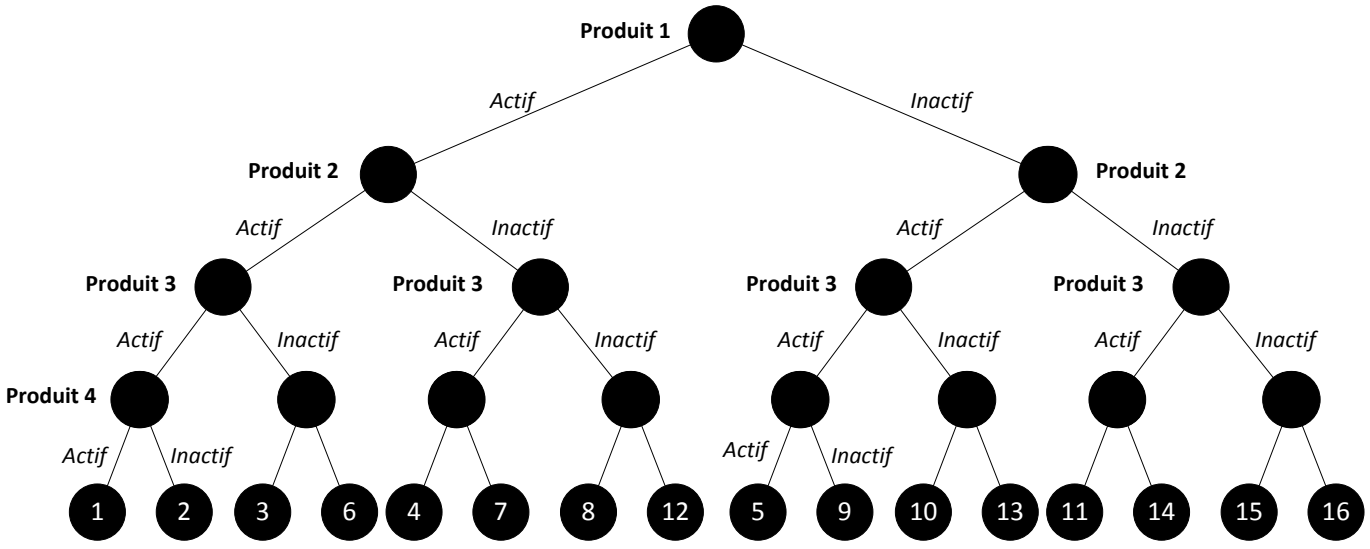

Figure 5. Ordre des feuilles visitées par LDS pour un arbre de profondeur 4

6 EXPERIMENTATION

Nous avons eu accès aux données d'une usine de bois de plancher utilisant un système de découpe tel que décrit dans la section 4. Nous disposons des scans d'un lot de 3340 planches passées qui nous permettent de réaliser nos simulations d'entraînement. Nous disposons de 100 heures pour réaliser nos simulations ce qui veut dire que nous n'avons le droit que de tester 298 configurations. Par la suite, $G\left(\{S(x,\xi)\} \mid x \in \bar{X}\right)$ est utilisé pour produire un plan qui sera exécuté pour un lot de 6680 planches différentes de celles utilisées à l'entraînement.

Puisque les planches sont différentes de celles utilisées à l'entraînement, les contraintes de marché peuvent parfois ne pas être respectées. Étant donné que la production réelle diffère de celle prévue par le plan, il est possible de rouler *G* à nouveau pendant l'exécution pour rectifier cet écart. Deux méthodes d'application du plan différentes sont ainsi évaluées. La première, le plan est simulé sur l'ensemble du lot, la seconde, le plan est réactualisé à intervalles réguliers pour prendre en compte les éventuels écarts entre la production réelle et la production prévue.

Pour plus de détails sur le fonctionnement du système de production de bois de plancher et du simulateur utilisé, le lecteur pourra se référer à Wery et al. (2015).

Dans l'expérimentation qui suit, nous allons comparer 3 approches pour choisir les configurations à tester : ad hoc, DFS et LDS. La première est basée sur la connaissance des configurations utilisées lors de précédentes productions/simulations. Dans le cas de DFS et LDS nous comparerons également l'impact de différentes stratégies d'énumération des produits (i.e. comment déterminer lesquels placer en haut de l'arbre). Nous évaluerons les stratégies suivantes :

- Produits disposés au hasard
- Produit le plus rentable en premier
- Produit le moins rentable en premier
- Produit le plus produit (par rapport à des productions passées) en premier
- Produit le moins produit (par rapport à des productions passées) en premier

Nous disposons de 165 produits différents que l'usine peut produire. Tout d'abord, un filtrage a été effectué pour ne pas évaluer des combinaisons impossibles ou improbables. Le filtrage nous permet donc de ne nous occuper que des produits pouvant changer d'états qui sont au nombre de 24. Le nombre de combinaisons possibles passe alors à 2^{24}.

Pour chaque couple stratégie de recherche/stratégie d'énumération des produits, nous obtenons donc un « historique de production » associé. Cela nous permet de déterminer un plan grâce au modèle de planification.

Chaque plan est ensuite appliqué sur l'ensemble de tests. Pour chaque scénario, 30 réplications sont effectuées. À chaque réplication, l'ordre des planches est modifié. Néanmoins, la découpe des planches est simulée dans le même ordre pour chaque $i^{ème}$ réplication d'un scénario (i.e. pour chaque scénario, les planches sont découpées dans le même ordre pour la $i^{ème}$ réplication, ce qui satisfait à la condition des *Common Random Numbers*).

Une heuristique ad hoc est utilisée comme point de comparaison. Cette heuristique a été élaborée spécialement pour trouver de « bonnes » configurations (listes de produits actifs) pour le problème appliqué au bois de plancher. Elle se base sur des résultats de simulations préalablement effectuées. Une fois ces listes déterminées, la méthode décrite précédemment est utilisée pour les phases suivantes.

7 RESULTATS

Le tableau 1 présente la valeur monétaire moyenne obtenue en fonction de l'approche utilisée ainsi que la demi-largeur de l'intervalle de confiance à 95% pour 30 réplications. À noter qu'à des fins d'interprétation nous présentons la valeur annualisée de la production.

Dans le cadre de l'algorithme LDS, nous trouvons de bonnes solutions, viables, et ceci dans le temps imparti. Sans replanification, l'algorithme ad hoc est un peu meilleur (+0,032 %) que LDS. Avec replanification, il n'y a pas de différence significative. Entre les différentes heuristiques de choix de variables, il n'est pas possible de dénoter des différences significatives. Il semble donc que les résultats ne dépendent pas de la stratégie d'énumération des produits. En effet, pour environ 300 combinaisons (listes de produits actifs) différentes, l'arbre de recherche est à la fin de la seconde déviation. De ce fait, les feuilles visitées se ressemblent énormément et seules quelques feuilles sont différentes. Cela explique le fait que ce sont majoritairement toujours les mêmes combinaisons qui sont utilisées et que l'heuristique n'a pas (sans replanification) ou peu (avec replanification) d'impacts sur le résultat.

LDS semble donc être une bonne approche, car les résultats obtenus sont semblables à l'heuristique ad hoc pour le bois de plancher. Néanmoins, cette dernière ne peut fonctionner dans le cas d'un design d'un réseau d'usines alors que l'approche LDS peut s'y appliquer. C'est donc une approche généralisable qui ne requiert pas de connaissances particulièrement poussées du problème. Elle s'avère être rapide et simple d'implémentation.

À l'inverse, pour DFS, l'heuristique de recherche a un impact très fort puisque ce sont toutes les combinaisons autour d'une feuille qui vont être vérifiées. Nous aurons donc des combinaisons plus similaires qui vont être testées pour une heuristique. De ce fait, il n'a pas été possible de trouver une solution satisfaisant les contraintes du modèle. En effet, après 300 itérations de DFS (soit 300 combinaisons de produits actifs/inactifs différentes), sur les 24 produits, seuls neuf produits ont été au moins une fois inactivés. Les combinaisons ne sont donc pas représentatives. Dans le cadre de notre étude, DFS est donc à proscrire.

Tableau 1. Résultats des différentes optimisations/simulations pour chaque couple Algorithme/Heuristique

Algorithme	Stratégie d'énumération des produits	Replanification	
		Sans	Avec
		Valeur annuelle	Valeur annuelle
DFS	Quelconque	- -	- -
	Plus rentable	- -	- -
	Moins rentable	- -	- -
	Plus produit	- -	- -
	Moins produit	- -	- -
LDS	Quelconque	40 351 539 ± 11 632	40 492 468 ± 7 098
	Plus rentable	40 351 539 ± 11 632	40 492 896 ± 6 856
	Moins rentable	40 351 539 ± 11 632	40 493 069 ± 6 863
	Plus produit	40 351 539 ± 11 632	40 493 332 ± 6 922
	Moins produit	40 351 539 ± 11 632	40 492 960 ± 7 159
Ad-hoc		40 364 506 ± 9 602	40 491 317 ± 8 731

8 CONCLUSION ET PERSPECTIVES

Dans le cadre de ce projet, nous avons défini formellement le problème de la recherche de la meilleure combinaison de scénarios. Pour résoudre ce problème, des algorithmes d'optimisation combinatoire sont utilisés sur un cas d'étude appliqué à la découpe du bois de plancher. Le but est de trouver le meilleur plan possible dans un contexte où le temps nous manque. Il s'avère que l'algorithme DFS n'est pas adapté à notre contexte. L'autre algorithme étudié, LDS, fourni de bien meilleurs résultats. Étant donné le nombre élevé de combinaisons possibles et le nombre de combinaisons différentes relativement petit, en comparaison, qu'il a été possible de simuler (300 pour chaque couple algorithme/stratégie d'énumération des produits), nous n'avons pas noté de différences significatives entre les différentes stratégies d'énumération des produits.

Dans nos travaux futurs, il serait intéressant de comparer LDS avec d'autres algorithmes tels que DDS ou des méthodes dynamiques. De plus, les heuristiques de choix de variables avec LDS donnent des résultats identiques (ou similaires). En effet, le nombre de feuilles à visiter durant le temps imparti était tel que les feuilles visitées étaient presque toujours les mêmes. Il serait donc pertinent d'évaluer l'impact d'une augmentation (ou diminution) de ce nombre sur les solutions pour déterminer l'importance du choix de cette heuristique.

9 REFERENCES

Andradóttir, S. (2006). An Overview of Simulation Optimization via Random Search. In *Handbooks in Operations Research and Management Science*, pp. 617-631. Elsevier.

Andradóttir, S. (2015). A Review of Random Search Methods. In *Handbook of Simulation Optimization*, pp. 277-292. New York, NY: Springer New York.

April, J., Glover, F., Kelly, J.P., Laguna, M. (2003). Practical introduction to simulation optimization. *2003 Winter Simulation Conference*, 7-10 Décembre.

Barták, R. (2004). Incomplete depth-first search techniques: A short survey. *Proceedings of the 6th Workshop on Constraint Programming for Decision and Control, Ed. Figwer J.*

Barton, R.R., Meckesheimer, M. (2006). Metamodel-Based Simulation Optimization. In *Handbooks in Operations Research and Management Science*, pp. 535-574. Elsevier.

Better, M., Glover, F., Kochenberger, G., Wang, H. (2008). Simulation optimization: Applications in risk management.

International Journal of Information Technology & Decision Making, 7(4), pp. 571-587.

Carson, Y., Maria, A. (1997). Simulation Optimization: Methods And Applications. *The 1997 Winter Simulation Conference*, 7-10 Décembre.

Chen, C.-H., Chick, S.E., Lee, L.H., Pujowidianto, N.A. (2015). Ranking and selection: Efficient simulation budget allocation. In *Handbook of Simulation Optimization*, pp. 45-80. Springer New York.

Fu, M.C. (2002). Optimization for simulation: Theory vs. Practice. *INFORMS Journal on Computing*, 14(3), pp. 192-215.

Fu, M.C. (2015). Handbook of simulation optimization. Vol. 216: Springer New York.

Fu, M.C., Glover, F.W., April, J. (2005). Simulation optimization: a review, new developments, and applications. *The 2005 Winter Simulation Conference*, 4-7 Décembre.

Goldsman, D., Nelson, B.L. (1994). Ranking, selection and multiple comparisons in computer simulation. *The 1994 Winter Simulation Conference*, 11-14 Décembre.

Harvey, W.D., Ginsberg, M.L. (1995). Limited discrepancy search. *Ijcai-95 - Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, San Mateo.

Jian, N., Henderson, S.G. (2015). An introduction to simulation optimization. *The 2015 Winter Simulation Conference*, 6-9 Décembre.

Kelton, W.D., Barton, R.R. (2003). Experimental design for simulation. *The 2003 Winter Simulation Conference*, 7-10 Décembre.

Kiefer, J., Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, pp. 462-466.

Kim, S., Pasupathy, R., Henderson, S.G. (2015). A Guide to Sample Average Approximation. In *Handbook of Simulation Optimization*, pp. 207-243. New York, NY: Springer New York.

Kleijnen, J.P.C. (2015). Response Surface Methodology. In *Handbook of Simulation Optimization*, pp. 81-104. New York, NY: Springer New York.

Kleijnen, J.P.C., Sanchez, S.M., Lucas, T.W., Cioppa, T.M. (2005). State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing*, 17(3), pp. 263-289.

Korf, R.E. (1996). Improved limited discrepancy search. *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, Oregon, August 4–8, 1996.

Ladier, A.-L., Greenwood, A.G., Alpan, G.u.u., Hales, H. (2014). Issues in the complementary use of simulation and optimization modeling. In *Les Cahiers Leibniz*.

Law, A.M. (2007). Simulation modeling and analysis. 4th ed ed. Boston: McGraw-Hill.

Robbins, H., Monro, S. (1951). A Stochastic Approximation Method. *The annals of mathematical statistics*, pp. 400-407.

Shapiro, A. (2013). Sample Average Approximation. In *Encyclopedia of Operations Research and Management Science*, pp. 1350-1355. Boston, MA: Springer US.

Wery, J., Marier, P., Gaudreault, J., Chabot, C., Thomas, A. (2015). Improving a hardwood flooring cutting system through simulation and optimization. *The 2015 Winter Simulation Conference*, 6-9 Décembre.