AMÉLIORATIONS AUX SYSTÈMES À INITIATIVE PARTAGÉE HUMAIN-ORDINATEUR POUR L'OPTIMISATION DES SYSTÈMES LINÉAIRES

Mémoire

François Chéné

Maîtrise en informatique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Résumé

La programmation linéaire permet d'effectuer l'optimisation de la gestion des réseaux de création de valeur. Dans la pratique, la taille de ces problèmes demande l'utilisation d'un ordinateur pour effectuer les calculs nécessaires, et l'algorithme du simplexe, entre autres, permet d'accomplir cette tâche.

Ces solutions sont cependant construites sur des modèles approximatifs et l'humain est généralement méfiant envers les solutions sorties de « boîtes noires ». Les systèmes à initiative partagée permettent une synergie entre, d'une part, l'intuition et l'expérience d'un décideur humain et, d'autre part, la puissance de calcul de l'ordinateur.

Des travaux précédents au sein du FORAC ont permis l'application de cette approche à la planification tactique des opérations des réseaux de création de valeur. L'approche permettrait l'obtention de solutions mieux acceptées. Elle a cependant une interface utilisateur limitée et contraint les solutions obtenues à un sous-espace de l'ensemble des solutions strictement optimales.

Dans le cadre de ce mémoire, les principes de conception d'interface humain-machine sont appliqués pour concevoir une interface graphique plus adaptée à l'utilisateur type du système. Une interface basée sur le modèle de présentation de données de l'outil Logilab, à laquelle sont intégrées les interactivités proposées par Hamel *et al.* est présentée.

Ensuite, afin de permettre à l'expérience et à l'intuition du décideur humain de compenser les approximations faites lors de la modélisation du réseau de création de valeur sous forme de problème linéaire, une tolérance quant à l'optimalité des solutions est introduite pour la recherche interactive de solutions alternatives. On trouvera un nouvel algorithme d'indexation des solutions à combiner et une nouvelle heuristique de combinaison convexe pour permettre cette flexibilité.

Afin d'augmenter la couverture de l'espace solutions accessible au décideur humain, un algorithme de recherche interactive de solution basé sur le simplexe est introduit. Cet algorithme présente une stabilité similaire à la méthode de Hamel *et al.*, mais ses performances en temps de calcul sont trop basses pour offrir une interactivité en temps réel sur de vrais cas industriels avec les ordinateurs présentement disponibles.

Une seconde approche d'indexation complète de l'espace solutions est proposée afin de réduire les temps de calcul. Les nouveaux algorithmes « Linear Redundancyless Recursive Research » (Recherche linéaire récursive sans redondance, LRRR) pour la cartographie et l'indexation de l'espace solutions et « N-Dimension Navigation Direction » (direction de navigation à n-dimensions, NDND) pour l'exploration interactive de celui-ci sont présentés. Ces algorithmes sont justes et rapides, mais ont cependant un coût mémoire au-delà de la capacité des ordinateurs contemporains.

Finalement, d'autres pistes d'exploration sont présentées, notamment l'exploitation des méthodes du point intérieur et de l'algorithme de Karmarkar ainsi qu'une ébauche d'approche géométrique.

Table des matières

Liste des figures

Liste des tableaux

Remerciements

J’aimerais remercier mon directeur de recherche, monsieur Jonathan Gaudreault et mon codirecteur, monsieur Claude-Guy Quimper, qui m’ont tous deux guidé tout au long de mes recherches.

J’aimerais aussi remercier Francine Lorrain pour la correction linguistique du présent document.

Je remercie enfin l’équipe du Consortium de recherche FORAC.

Avant-propos

Ce mémoire présente un travail de recherche qui s'inscrit dans la continuité des travaux précédemment réalisés par messieurs Simon Hamel, Jonathan Gaudreault, Claude-Guy Quimper, Mathieu Bouchard, et Philippe Marier du consortium de recherche FORAC [1] sur l'utilisation des systèmes à initiative partagée dans le contexte de l'optimisation d'un réseau de création de valeur dans l'industrie forestière.

Le présent ouvrage présente les différentes améliorations apportées aux méthodes développées précédemment au sein du Consortium de recherche FORAC.

Une partie des résultats a été publiée dans un article de conférence avec comité de lecture (« A mixed-initiative system for interactive tactical supply chain optimization », *10ème Conférence Francophone de Modélisation, Optimisation et Simulation : de l'économie linéaire à l'économie circulaire*, 5-7 novembre 2014), article dont je suis premier auteur et pour lequel j'ai réalisé les expérimentations. Les coauteurs sont mon directeur et mon codirecteur de recherche.

Ces résultats ont également été rapportés dans un article de journal (« Designing a Generic Human-Machine Framework for Real-time Supply Chain Planning », *Journal of Computational Design and Engineering*, 30 décembre 2016) dont je suis coauteur.

Introduction

Dans « Origins of the Simplex Algorithm », George Dantzig rapporte que bien que la programmation linéaire soit bien connue depuis le XIX^{e} siècle, elle n'a pris son essor comme domaine d'étude qu'avec l'arrivée de machines de calcul rapides et puissantes à la sortie de la Seconde Guerre mondiale. En effet, s'il est possible d'utiliser la programmation linéaire pour représenter un problème de planification d'opération et d'obtenir un plan (ou programme) optimal, la quantité de données à gérer et de calculs à effectuer pour un cas réel place cet exercice hors de la portée d'un humain.

La planification optimale des tâches au sein d'un réseau de création de valeur fait partie des problèmes pouvant être résolus par la programmation linéaire grâce à des algorithmes, comme celui du simplexe, exécutés sur des ordinateurs. L'approche a cependant des limites. Tout d'abord, la modélisation d'un réseau de création de valeur doit être faite par un humain qui peut faire des erreurs et doit souvent faire des approximations. De plus, la quantité de données à gérer et de calculs à effectuer est telle que même un ordinateur moderne nécessite un temps de calcul qui se mesure en heures pour obtenir une solution. La correction d'une erreur de modélisation découverte lors de la lecture de la solution demande donc du temps. Finalement, les êtres humains sont naturellement méfiants envers les solutions sorties comme par magie d'une machine.

Ces limites peuvent conduire une entreprise à se tourner vers un expert humain qui fera appel à son expérience passée et son intuition pour concevoir un plan d'opération à peu près optimal. Il n'y a cependant aucune garantie quant à l'optimalité des plans obtenus et la qualité de ceux-ci dépend de la qualité de l'expertise de la personne.

Les systèmes à initiative partagée offrent une solution pour combiner les forces et repousser les limites des deux approches. Ces systèmes permettent une synergie entre l'intuition et l'expérience d'un décideur humain et la puissance de calcul

logique de l'ordinateur pour effectuer des tâches [2]. Les travaux antérieurs réalisés au sein du Consortium de recherche FORAC ont ouvert une piste prometteuse pour appliquer cette approche dans le contexte de l'optimisation des systèmes linéaires, avec une application à la planification tactique des opérations dans un réseau de création de valeur [1].

Si l'approche est prometteuse, le système résultant de ces travaux a cependant deux limites importantes. D'une part, le modèle de présentation ne place pas les données en contexte et demande donc à l'humain collaborant à la tâche d'utiliser une partie de ses capacités pour comprendre la réalité associée à chaque valeur et à mettre celles-ci en relation. D'autre part, le système ne permet à la personne que d'explorer des solutions jugées parfaitement optimales par l'ordinateur, et une partie de ses solutions ne sont pas accessibles.

Le but des travaux rapportés dans le présent mémoire est donc l'amélioration du système en permettant à l'humain de mieux mettre à profit son intuition et son expertise dans l'élaboration d'un plan tactique d'opération.

Dans le chapitre 1, les notions mathématiques et informatiques nécessaires à la compréhension du problème étudié sont présentées. On y trouve d'abord le détail des principes de la programmation linéaire et de l'algorithme du simplexe. Les systèmes à initiative partagée et leur application à la planification tactique des opérations dans le cadre de l'industrie forestière y sont ensuite décrits. Le système proposé par Hamel et *al* y est ensuite expliqué. Les principes de conception d'une interface personne-machine y sont ensuite exposés, suivis d'un modèle de présentation de données utilisé comme base dans la suite des travaux. Cette section se termine avec la notion d'instabilité numérique, une limite importante au calcul massif effectué lors de la résolution d'un problème linéaire.

Le chapitre 2 propose une nouvelle interface graphique humain-machine pour un système à initiative partagée d'optimisation de réseaux de création de valeur. On y

détaille le processus itératif de conception et l'explication des choix technologiques en soutenant le fonctionnement.

Le chapitre 3 présente l'introduction d'une tolérance à la quasi-optimalité pour l'exploration interactive de solutions. On y détaille les ajustements à réaliser à la méthode de Hamel *et al.* Une nouvelle heuristique de combinaison de solutions et un nouvel algorithme d'indexation des solutions à combiner adaptés à cette tolérance sont présentés, suivis d'un protocole expérimental et des résultats validant ceux-ci.

Dans le chapitre 4, une version interactive de l'algorithme du simplexe pour tenter de couvrir la totalité de l'espace solutions est proposée. L'élaboration d'une nouvelle heuristique de pivot et de l'algorithme associés y est détaillée. On y retrouve finalement le protocole expérimental et les résultats obtenus pour la mesure des performances de cette méthode.

Le chapitre 5 propose une approche de cartographie complète de l'espace solutions pour accélérer la génération interactive de solutions. Un algorithme de navigation dans un nuage de solutions quelconque y est proposé, avec un protocole expérimental et les résultats obtenus pour celui-ci. Un algorithme de cartographie de la totalité de l'espace solutions y est introduit, lui aussi accompagné d'un protocole expérimental et de résultats.

La conclusion présente une synthèse des résultats obtenus pour les différentes méthodes avancées et testées pour l'amélioration d'un système à initiative partagée pour l'optimisation d'un réseau de création de valeur. On y trouve aussi quelques pistes explorées en surface durant les travaux rapportés.

1.Notions préliminaires

Afin de comprendre correctement les travaux rapportés dans ce mémoire, il convient de se familiariser d'abord avec quelques notions. Ces notions sont : les systèmes linéaires, notamment leur optimisation par l'application de l'algorithme du simplexe et leur convexité, l'instabilité numérique, les réseaux de création de valeur et les systèmes à initiative partagée.

1.1. Programmation linéaire

Une ***variable*** est une donnée d'un modèle mathématique pour laquelle la valeur n'est pas fixée.

Le ***domaine*** d'une variable correspond à l'ensemble des valeurs pouvant être attribuées à celle-ci.

Une ***contrainte*** est une limite formelle sur l'ensemble des valeurs faisant partie du domaine de variables. Elle peut s'appliquer directement à une seule variable ou dépendre d'une relation entre plusieurs variables.

Un ***problème linéaire*** est l'application d'un ensemble de contraintes sous forme d'équations linéaires (du premier degré) à un ensemble de variables dont les valeurs sont des nombres réels non négatifs [3].

Une ***solution*** à un problème linéaire est un ensemble de valeurs attribuées aux variables du modèle. Si une solution respecte toutes les contraintes, elle est dite *réalisable*. On appelle *problème de satisfaction linéaire* la recherche d'une solution réalisable à un problème linéaire [3].

Par exemple, soit un problème à trois (3) variables $x, y, z \in [0,10]$. Ajoutons une contrainte stipulant que la somme des variables doit être inférieure à 10. Cette contrainte s'exprime par l'inéquation $x + y + z \leq 10$. Pour un tel programme, la

solution $x = 7, y = 7, z = 7$ n'est pas réalisable car $x + y + x = 21 \nleq 10$. En revanche, la solution $x = 0, y = 0, z = 0$ est réalisable.

Plusieurs problèmes réels peuvent être modélisés sous la forme d'un problème de satisfaction linéaire. La recherche d'un plan tactique d'opération réalisable en fonction des contraintes posées par les limites physiques, légales et économiques d'un réseau industriel en est un exemple [1]. L'une des premières applications concrètes de l'optimisation linéaire ayant été l'établissement de programmes d'opération, on parle parfois de ***programmation linéaire*** et on réfère à la solution comme à un ***programme***.

Un ***problème d'optimisation*** consiste à trouver, parmi les solutions réalisables d'un problème de satisfaction, une solution telle qu'une fonction f sur une, plusieurs ou la totalité des variables du problème soit la plus petite (ou la plus grande) possible. On appelle cette fonction la « *fonction-objectif* ». On parle de *minimisation* lors de la recherche d'une solution ayant la plus petite valeur possible et de *maximisation* pour la recherche d'une solution ayant la plus grande valeur possible. On peut obtenir un problème d'optimisation à partir d'un problème de satisfaction par l'ajout d'une fonction à optimiser [3].

Bien que ces problèmes puissent s'exprimer clairement par des nombres réels ($\mathbb{R}$) et des équations linéaires simples, les problèmes linéaires industriels ont un très grand nombre de variables et de contraintes [1]. De plus, les contraintes reliant les variables impliquent un changement des domaines de celles-ci à chaque tentative d'assignation de valeur, ce qui rend la recherche de solutions réalisables et optimales complexe [4].

1.2. L'algorithme du simplexe

Introduit en 1947 par G. Dantzig [5], ***l'algorithme du simplexe*** est une méthode pour résoudre les problèmes d'optimisation sur un problème linéaire [3]. Bien qu'il soit non polynomial, il se révèle très performant dans la pratique. Pour comprendre

son fonctionnement, il convient de comprendre deux concepts principaux : la ***solution de base*** et le ***pivot***.

L'algorithme du simplexe requiert un programme sous sa ***forme standard***. Cette forme est présentée à l'équation (1). Or, tout problème d'optimisation linéaire peut être exprimé sous cette forme [4].

$$\begin{matrix} \min c^T x \\ Ax = b \\ x \geq 0 \end{matrix} \quad (1)$$

Soit un programme linéaire L à n variables et m contraintes. Une ***solution de base*** est une solution réalisable se trouvant à la limite de n contraintes [4]. L'algorithme du simplexe cherche une solution optimale parmi les solutions de base. À partir d'une première solution réalisable, il effectue un pivot, c'est-à-dire qu'il change la valeur attribuée à l'une des variables formant la ***base***. La base correspond à un groupe de m variables telles que, en leur assignant des valeurs pouvant être non nulles, on obtient une solution de base. Les variables qui ne sont pas dans la base ont une valeur de 0.

En pratique, l'algorithme organise les contraintes en une matrice A de taille $m \times n$ où A_{ij} correspond au coefficient sur x_j de la i^{e} contrainte. Il organise b en une matrice colonne et c en matrice rangée, puis il assemble le tout dans la matrice I présentée en (2), où I est la matrice diagonale unitaire de taille $m \times m$ et v la valeur de cx. La matrice S ajoute un ensemble de variables de relaxation pour transformer les inégalités de la forme $A_i x_i \leq b_i$ sous la forme$A_i x_i + s_i = b_i$.

$$L = \begin{bmatrix} c & 0 & v \\ A & I & b \end{bmatrix} \quad (2)$$

Dans cette structure, pour une variable x_j, si celle-ci appartient à la base, $c_j = 0$ et tous les $A_{ij} = 0$ sauf un qui vaut 1. La valeur de $x_j = A_j^T b_i$ si x_j appartient à la base sinon $x_j = 0$. Soit P une matrice de taille $m \times (m+n)$, la jonction $A|S$, C une matrice linéaire de taille $1 \times (m+n)$ la jonction $c|0$ et X la jonction $x|s$. L'opération de pivot consiste à faire entrer un X_j dans la base en effectuant une série d'opérations élémentaires sur les lignes de L afin que pour un élément P_{Ij}, $P_{Ij} = 1, \forall i \neq I$ tel que $P_{ij} = 0, c_I = 0$.

La clé de l'algorithme du simplexe est l'heuristique de choix du point de pivot. On choisit l'élément P_{Ij} tel que c_J soit plus petit que 0 et soit le plus petit élément de c et $\frac{b_I}{P_{Ij}}$ soit le plus petit parmi les $\frac{b_i}{P_{ij}}$ possibles, en résolvant les égalités par la plus petite valeur absolue de P_{ij}. Cette heuristique permet de choisir en temps linéaire $O(n+m)$ la plus grande réduction de v, ce qui fait de l'algorithme du simplexe un algorithme vorace de descente de gradient.

1.3. Géométrie de l'espace de solution

Les modèles linéaires ont des propriétés géométriques utiles pour la résolution du problème de satisfaction ou d'optimisation qui y est associé. Cette sous-section présente la correspondance géométrique d'un espace de solution à un modèle linéaire et de l'intérêt de sa convexité.

Correspondance géométrique d'un espace de solution

Un ***polytope*** est formé dans un (hyper)[1] espace euclidien par un sous-espace fermé, délimité par des segments de (hyper) demi-plans. On parle communément

[1] En géométrie euclidienne, on ajoute le suffixe hyper- au nom d'une structure pour dénoter qu'elle a plus de trois (3) dimensions.

de polygone dans un espace à deux (2) dimensions et de polyèdre dans un espace à trois (3) dimensions.

Soit un modèle linéaire sur n variables avec m contraintes. On peut projeter ce problème sur un espace euclidien à n dimensions orthogonales. Dans cet espace, chaque combinaison de valeurs attribuées aux variables du programme est un point de cet espace. Chaque contrainte A_i forme un demi-espace $A_i x \leq b_i$ délimité par un (hyper) plan $A_i x = b_i$ séparant deux demi-espaces, l'un contenant les solutions respectant la contrainte et l'autre les solutions invalides. Un espace solutions forme donc un polytope.

On appelle ***sommet*** l'intersection d'un nombre de plans égal à ou plus grand que le nombre de dimensions à la limite du polytope.

Un ***point extrême*** est une solution d'un espace de solutions linéaires située sur un sommet du polytope formé par l'espace solutions, c'est-à-dire une solution réalisable située à la limite d'un nombre de contraintes égal au nombre de variables. L'algorithme du simplexe effectue un parcours de points extrêmes pour trouver une solution optimale.

Convexité

Soit un espace S englobant un ensemble de points. On dit que S est ***convexe*** si pour toute paire de points x, y dans S et pour n'importe quelle valeur de α entre 0 et 1, le point obtenu par interpolation de x et y ($\alpha x + (1 - \alpha) y$) est aussi inclus dans S [4].

Lors de la correspondance d'un espace solutions à un polytope, il est à noter qu'un demi-plan sépare deux espaces convexes. L'espace solutions est l'intersection des demi-espaces valides. Or, l'intersection d'espaces convexes est aussi convexe [4]. L'ensemble des solutions réalisables forme donc un espace convexe.

Puisqu'un problème de satisfaction linéaire est convexe. Il ne présente pas d'optimum local. Par conséquent, les algorithmes de descente de gradient, tel que le simplexe, sont garantis d'être corrects [6].

Dégénérescence

On parle de ***dégénérescence*** lorsque plus de n contraintes se croisent au même sommet [3]. Dans cette situation, les bases ne correspondent pas aux sommets du polytope. Plus précisément, plusieurs bases correspondent au même sommet. Par conséquent, l'opération de pivot peut « bloquer » puisqu'un changement de base ne change pas nécessairement la solution courante [7].

1.4. Instabilité numérique

L'***approximation*** consiste à utiliser une valeur proche d'une valeur réelle pour la représenter. Il s'agit d'une pratique courante en science et en ingénierie où les précisions des mesures sont souvent limitées [8].

Les ordinateurs binaires encodent les nombres à virgule flottante, ou nombres réels ($\mathbb{R}$), sur des bits, une mémoire binaire. Cela implique une certaine approximation compte tenu de la limite de précision de cette structure discrète pour représenter une entité au domaine continu [9]. Par exemple, la valeur de $\frac{2}{3}$, ou $0,\overline{6}$, a un nombre infini de décimales. Or, les nombre à virgule flottante utilisés par les ordinateurs ont une limite de décimales pouvant être mémorisées. Un nombre à virgule à simple précision, avec ces 7 chiffres significatifs, doit arrondir la valeur à 0.6666667.

Ces erreurs peuvent avoir un impact sur les systèmes linéaires. J. R Bunch [8] présente un exemple sur un calcul linéaire simple. Soit $Ax = b$, l'approximation à 3 nombres significatifs de $\tilde{A}\tilde{x} = \tilde{b}$.

Soit $\tilde{A} = \begin{bmatrix} 1.00 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}$, $A = \begin{bmatrix} 1.00 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}$

Soit $\tilde{b} = \begin{bmatrix} 1.9899903 \\ 1.970106 \end{bmatrix}$, $b = \begin{bmatrix} 1.99 \\ 1.97 \end{bmatrix}$

Dans ce cas, $\tilde{x} = \begin{bmatrix} 3 \\ -1.0203 \end{bmatrix}$. Intuitivement, on penserait que $x = \begin{bmatrix} 3.00 \\ -1.02 \end{bmatrix}$. Or $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Une modification de l'ordre de 1.0×10^{-2} produit une erreur de 2!

Si ces imprécisions sont minimes, elles ont tendances à s'accumuler lors des calculs. Elles peuvent aussi affecter la transitivité et la commutativité [9]. Dans ces situations, deux séquences d'opérations mathématiquement équivalentes produiront des résultats différents lors de leur évaluation.

On appelle ce phénomène ***instabilité numérique***. L'instabilité numérique est problématique lors du parcours de points extrêmes car une même solution peut être évaluée pour des valeurs différentes si elles sont obtenues par des chemins différents, créant plus de points extrêmes qu'il n'en existe réellement.

On dit d'un polytope qu'il est ***perturbé*** lorsqu'un nombre plus grand de points extrêmes est trouvé suite par l'instabilité numérique. Kotiah et Steinberg [10] rapportent des cas particuliers où l'imprécision dans les calculs perturbe le polytope de l'espace solutions. Dans certains cas extrêmes, l'algorithme du simplexe entre dans des cycles lors de la recherche de la solution optimale et devient de fait incorrect puisqu'il ne converge jamais vers une solution.

Les implémentations d'algorithmes qui traitent des systèmes linéaires sur des ordinateurs réels doivent donc prendre cette réalité en considération.

1.5. Réseau de création de valeur

Le type de problèmes servant d'objet d'étude dans le cadre du présent projet est de type ***planification collaborative des ventes et de la production*** (***PCVP***), en anglais Sales & Operations Planning (S&OP) [11].

Un ***réseau de création de valeur*** comprend les sources d'approvisionnement (sites d'extraction de matière première ou fournisseurs externes), les unités de transformation et les points de sortie des produits transformés (ventes) d'un réseau de création de valeur ainsi que l'ensemble des moyens de transports des ressources et produits entre ceux-ci [12]. Comme mentionné précédemment, les quantités de travail effectué, de ressources consommées et de matière produite et transportée au sein d'un tel réseau, ainsi que l'ensemble des contraintes imposées par les relations entre ces quantités, peuvent être intégrés au sein d'un modèle linéaire afin d'en optimiser le rendement [13].

Un exemple simple de réseau de création de valeur est présenté à la figure 1. Dans cette figure, les nœuds représentent les sites d'exploitation, de transformation et de distribution des produits de la forêt tandis que les arcs représentent les axes de déplacement de ressources et de produits entre ceux-ci.

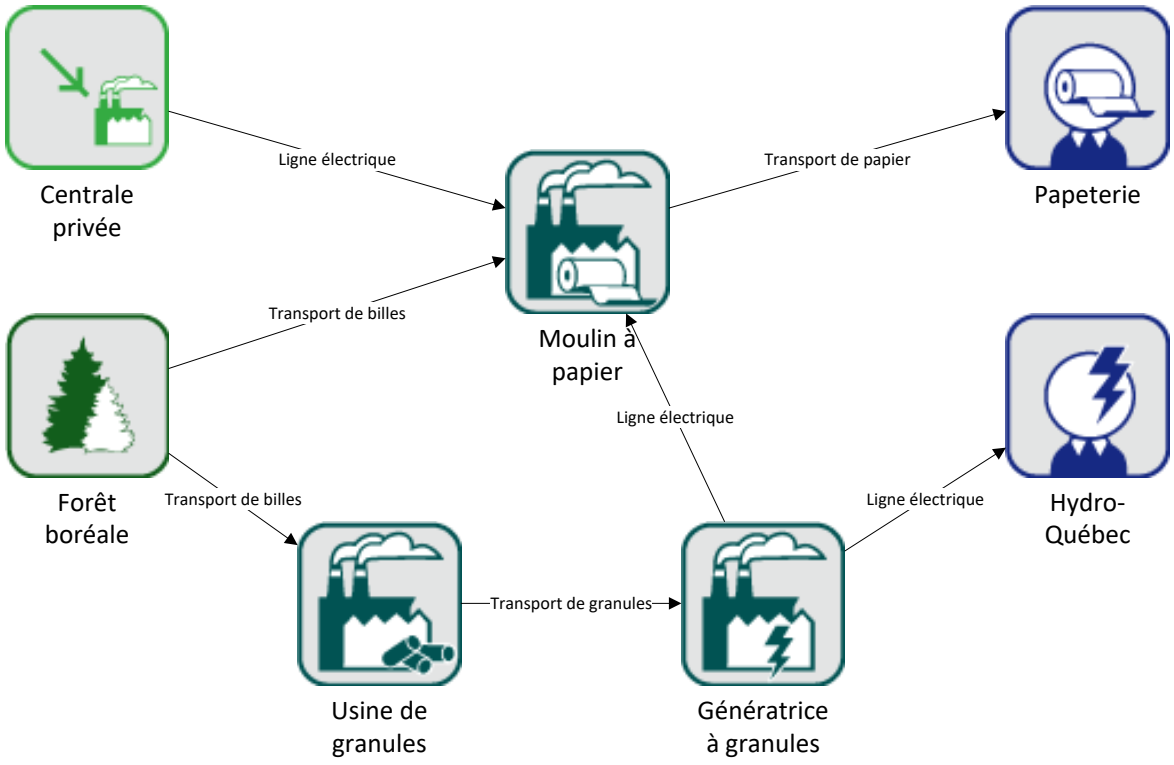

Figure 1 - Réseau de création de valeur simple

Dans ce réseau, un site de coupe (forêt boréale) produit des mètres cubes de billots de bois à chaque heure d'exploitation. Les billes produites peuvent être acheminées par camion, soit à une usine de granules, soit à un moulin à papier. L'usine de granules transforme des billes en granules de bois combustibles à chaque heure d'opération. Les granules produits sont acheminés par train à une génératrice. Cette usine brûle des granules pour produire de l'électricité à chaque heure d'opération. L'électricité produite est acheminée par ligne électrique soit au moulin à papier, soit à Hydro-Québec, un acheteur. Le moulin à papier transforme des billes de bois en tonnes de papier à chaque heure d'opération. Chaque heure d'opération demande l'utilisation de l'électricité provenant soit de la génératrice à pastille, soit achetée à un fournisseur externe. Le papier produit est expédié par camion à une papeterie qui achète le papier.

Un modèle du système linéaire du PCVP associé à ce réseau pour trois périodes de temps est présenté dans l'Annexe 1 : Modèles linéaires utilisés pour évaluer les performances des algorithmes au modèle 6.

La ***planification collaborative des ventes et de la production*** est un processus par lequel on synchronise les décisions de planification de production, de distribution et de vente dans le but de maximiser les profits générés par l'exploitation du réseau de création de valeur. Ce plan se fait sur un horizon de trois (3) à dix-huit (18) mois et intègre en un plan unifié les opérations de chaque branche de la firme afin d'optimiser le rendement de celle-ci [11].

1.6. Logilab

LogiLab [14] est un outil de PCVP qui présente le modèle d'un réseau de création de valeur sous la forme d'un graphe, les nœuds en étant les unités d'affaires (site de coupe, moulin à scie, client, etc…) et les arcs axes de transport que les ressources et les produits peuvent emprunter pour passer d'une unité d'affaires à une autre. Pour chaque unité d'affaires, le nom en est donné, et le taux de production planifié par rapport à la capacité de l'unité est affiché. Les arcs sont plus ou moins

larges selon le volume de produits transportés et leur couleur est fixée en fonction de la nature du produit transporté. La direction du déplacement est aussi indiquée par un chevron au milieu de l'arc. Pour l'ensemble du graphe, les données peuvent être présentées pour une période réduite ou pour l'ensemble de la durée couverte par l'exercice de planification. L'aspect de LogiLab est présenté dans la figure 2.

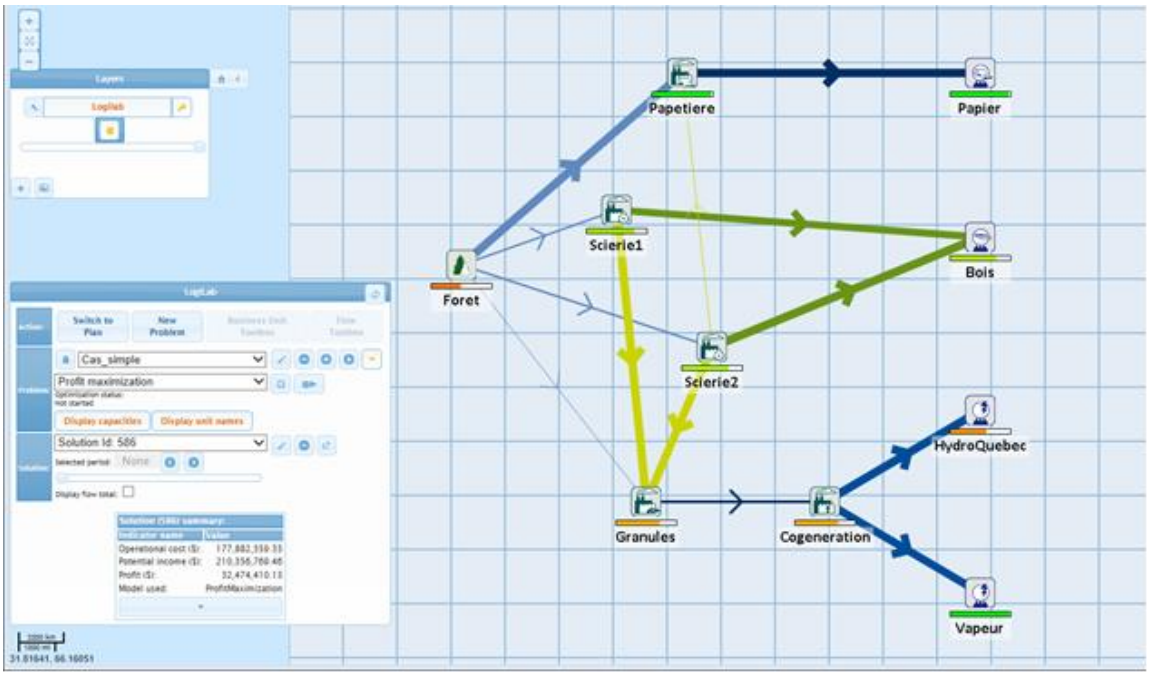

Figure 2 - L'outil LogiLab

1.7. Systèmes à initiative partagée

La résolution d'un problème d'optimisation requiert une grande quantité de calculs. Les processeurs numériques sont particulièrement bien adaptés à ce type de tâche. Cependant, l'humain possède une connaissance des problèmes ne pouvant pas toujours être formalisée [15].

Les ***systèmes à initiative partagée*** sont des systèmes où l'ordinateur et l'humain agissent à tour de rôle pour faire intervenir leurs compétences respectives dans la résolution d'un problème [2]. Ces systèmes présentent les avantages suivants :

- la direction humaine permet aux algorithmes de converger plus rapidement vers une solution optimale [15];
- la participation du décideur à l'élaboration de la solution augmente la compréhension et l'acceptation de celle-ci [15];
- le résultat obtenu par la collaboration est généralement de meilleure qualité que les résultats obtenus par un humain ou un ordinateur seul [16].

La majorité des recherches faites sur l'application des systèmes à initiative partagée se concentre plus sur l'optimisation combinatoire.

Kun et Havens [17] proposent un système à initiative partagée pour l'établissement d'un parcours académique. Ce système repose sur une interface graphique centrée sur la visualisation d'information et la manipulation directe par l'utilisateur pour une collaboration efficace. La recherche de solution s'y fait dans un premier temps par un solveur automatisé, puis dans une approche de collaboration humain-machine par initiative partagée, afin de satisfaire à la fois les impératifs académiques et les préférences de l'utilisateur. L'expérimentation sur des cas réels a produit des résultats positifs.

Ai-Chang et *al.*, Bresina et Morris [18], [19] et [20] proposent MAPGEN, un système à initiative partagée pour la planification des activités du Rover lors de l'exploration et de la collecte de données de Mars pour optimiser l'utilisation du temps et des senseurs du robot. Le système proposé est la fusion de deux systèmes existants, APGEN, un système de planification d'activités du « Jet Propulsion Laboratory » et le système de planification Europa du centre de recherche Ames de la NASA. Ce système a participé au succès de la mission Mars Exploration Rover

Guiost et *al.* [21] proposent un système à initiative partagée de contrôle du trafic aérien pour la collaboration entre deux opérateurs humains. Leurs travaux n'ont malheureusement pas permis de mesurer l'efficacité du système proposé.

Lenor et *al.* [22] proposent un système de planification d'itinéraire pour des convois militaires. Le système à initiative permet au commandant humain de modifier les paramètres des itinéraires et de la composition des convois. Dans tous les cas

analysés, les commandants assistés par le système ont mieux performé que les commandants du groupe de contrôle.

Hamel et *al.* [1] notent que l'approche par initiative partagée est peu abordée pour les systèmes linéaires puisque les algorithmes de résolutions existants comme le simplexe sont assez robustes et efficaces. Ils proposent tout de même un système à initiative partagée pour l'optimisation d'un modèle linéaire. Ce système fonctionne comme suit.

1. Un décideur identifie les métriques clés, incluant la fonction-objectif permettant d'évaluer une solution, pour l'exercice de planification.
2. Un expert modélise le réseau de création de valeur sous forme de problème d'optimisation linéaire.
3. Un solveur d'optimisation linéaire calcule la valeur optimale pour la fonction-objectif du modèle ainsi que la *zone d'optimalité* pour chacune des *variable clés* dont le décideur souhaite pouvoir modifier la valeur. La zone d'optimalité est la plage de valeurs que peut prendre cette variable pour laquelle il existe au moins solution optimale au problème.
4. Le système présente la première solution trouvée ainsi que la zone d'optimalité de chacune des *variables clés* préidentifiées par le décideur.
5. Le décideur peut alors ajuster les valeurs des variables clés (pour autant que la nouvelle valeur soit dans la zone d'optimalité) en suivant son expérience et sa compréhension du problème jusqu'à l'obtention d'une solution satisfaisante.
6. Après chaque manipulation, le système génère une nouvelle solution optimale suffisamment rapidement pour permettre une interaction en temps réel et en évitant de faire des « bonds » qui risqueraient de confondre le décideur.

Ce flot d'interactivité est illustré à la figure 3.

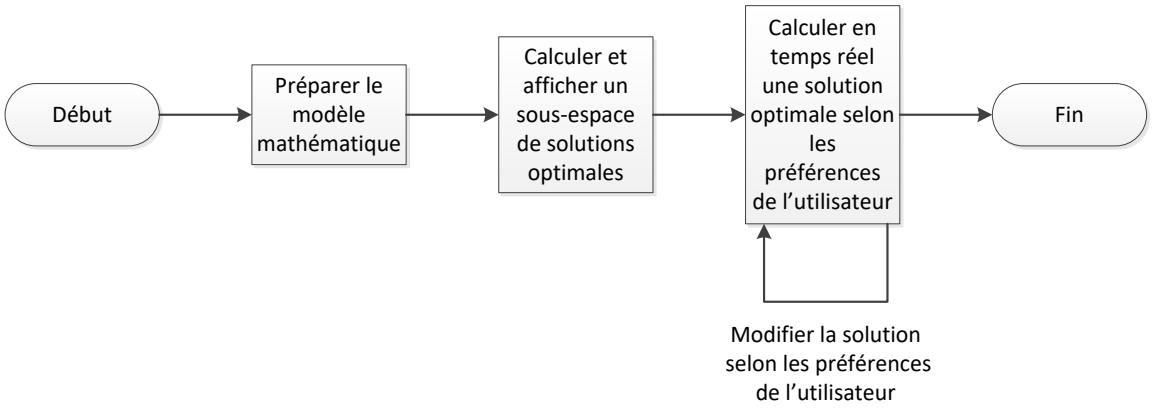

Figure 3- Processus interactif d'optimisation proposé par Hamel et al.

Les travaux de Hamel *et al.* [1] se concentrent principalement sur les algorithmes et les heuristiques soutenant le système à initiative partagé proposé. L'interface proposée est donc minimale (voir figure 4). Dans cet exemple, l'interface affiche les valeurs assignées à des quantités de matière transportées sur différents axes de transport, soit par train (en rouge), soit par camion (en bleu). Les valeurs de quantités sont présentées sous forme de diagramme à bandes verticales. Pour chacune des valeurs affichées, *la zone d'optimalité*, soit l'ensemble des valeurs pouvant être assignées à cette variable pour lesquelles il existe une solution optimale, est affiché sous la forme d'une barre verticale dont les extrémités sont mises en valeur par une barre horizontale de la largeur de la bande associée.

Bien que l'interface proposée par Hamel *et al.* présente l'ensemble des données pertinentes pour l'élaboration d'une solution, elle ne le fait pas dans le contexte visuel d'un réseau et n'illustre donc pas le lien entre ces différentes valeurs. Cette abstraction du contexte dans l'affichage ajoute à la charge mentale du décideur humain.

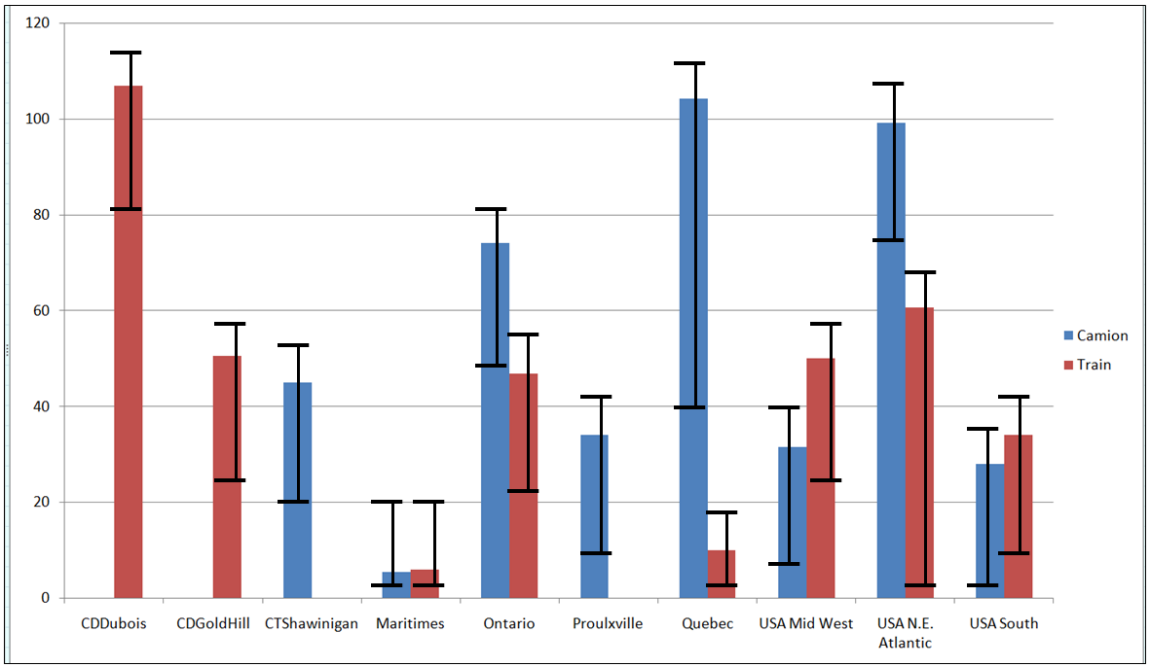

Figure 4- Proposition d'interface graphique affichant la zone d'optimalité de chaque valeur pour l'interaction humaine au sein d'un système à initiative partagée [1]

Heuristique triangulaire

Lors d'une manipulation sur la valeur d'une variable, le système génère une nouvelle solution optimale. Cette génération doit se faire suffisamment rapidement pour permettre une interaction en temps réel et éviter de faire des « bonds » qui risqueraient de confondre le décideur.

L'***heuristique triangulaire*** proposée par Hamel *et al.* [1] remplit ces critères. Elle fait une application de l'interpolation à des solutions d'un modèle linéaire pour obtenir de nouvelles solutions optimales dans un cadre d'interaction humain-machine. La prochaine sous-section décrit la méthode.

Application de l'interpolation aux solutions d'un modèle linéaire

Soit S l'ensemble des solutions réalisables à un problème linéaire L. Soit aussi x et y, deux solutions réalisables de L et donc incluses dans S. Puisque S est convexe,

toute interpolation de x et y est aussi incluse dans S, et donc une solution réalisable de L. On peut donc obtenir de nouvelles solutions à un problème linéaire à partir de solutions connues, et ce, en temps linéaire.

Obtention de solutions à un modèle linéaire par interpolation

Soit un problème linéaire L tel que $Ax \leq b$ sur lequel on veut optimiser cx. Soit v la valeur optimale de cx. Créons un nouveau problème linéaire L' ayant les mêmes variables et contraintes que L et pour lequel $cx = v$. La construction de A' est donnée à l'équation (3) :

$$A' : \{ \begin{array}{r} Ax \leq b \\ c \leq b \\ -c \leq -b \end{array} \} \quad (3)$$

L'ensemble des solutions réalisables de L' correspond à l'ensemble des solutions optimales de L. L' étant un modèle linéaire, il est convexe. L'ensemble des solutions optimales d'un programme linéaire est donc aussi un espace convexe [4]. On peut donc obtenir de nouvelles solutions optimales à partir de solutions optimales connues.

Application de l'interpolation de solution à l'interaction humain-machine

Cette propriété peut être utilisée par un utilisateur humain pour naviguer en temps réel dans l'espace d'optimalité d'un programme linéaire. Soit un problème linéaire L pour lequel on exprime l'espace d'optimalité par le problème L'. Soit aussi un ensemble de variables x qu'un utilisateur désire manipuler. On peut générer un ensemble de solutions optimales de L en maximisant et minimisant chacune des variables $x_i \in x$ sur L'. On peut ensuite naviguer dans l'espace d'optimalité de L en générant des solutions optimales par interpolation entre la solution courante et une solution connue [1]. Ce principe est illustré à la figure 5.

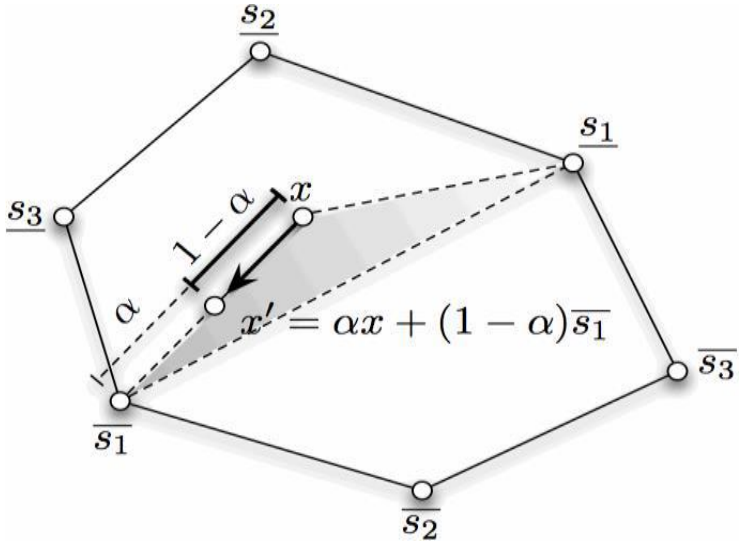

Figure 5 - Application de l'heuristique triangulaire. La nouvelle solution est calculée sans « saut » de valeurs par l'interpolation entre la solution courante et une solution extrême précalculée [1]

Plus précisément, la méthode proposée par Hamel *et al.* consiste à :

1. effectuer un calcul de la valeur optimale pour la fonction-objectif du modèle à l'aide d'un solveur classique (par exemple appliquant l'algorithme du simplexe);
2. ajouter au modèle une contrainte imposant l'égalité à la valeur obtenue pour la fonction d'optimisation;
3. pour chacune des variables pour lesquelles l'utilisateur pourra modifier la valeur attribuée, obtenir et indexer une solution minimisant et maximisant la valeur de cette variable;
4. lors de la modification de la valeur attribuée à une variable par l'utilisateur, utiliser l'index pour sélectionner la solution optimisant la valeur modifiée et faire la combinaison entre la solution affichée et la solution indexée pour en générer une nouvelle.

Cette méthode permet au système de générer des solutions optimales suivant les indications du décideur suffisamment rapidement pour que la collaboration soit faite dans un système en temps réel.

Cependant, cette méthode ne permet généralement que d'obtenir un sous-ensemble de l'espace solutions. Seules des solutions pour lesquelles une variable a une valeur à la limite de son domaine sont accessibles, et parmi ces solutions, une seule est conservée par variable. La figure 6 illustre cette limite :

certaines solutions sont laissées de côté, et en absence de combinaison convexe, il est impossible de parcourir l'espace avoisinant.

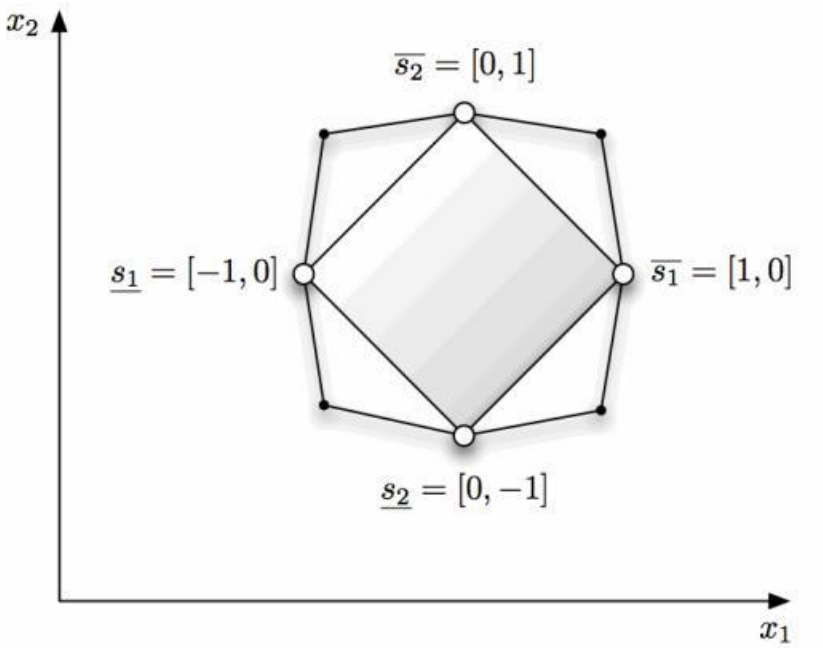

Figure 6 - couverture partielle de l'espace d'optimalité par l'algorithme de Hamel [1]

1.8 Conclusion

La programmation linéaire offre un outil pour la recherche de solutions et l'optimisation d'un problème où les valeurs attribuées à un ensemble de variables sont inconnues mais liées par un ensemble de contraintes. L'algorithme du simplexe est une méthode robuste et efficace pour résoudre ces problèmes. La rectitude de l'algorithme du simplexe repose en partie sur la convexité des espaces solutions d'un problème linéaire.

L'optimisation d'un réseau de création de valeur est un problème d'optimisation linéaire. Ces problèmes ont cependant un grand nombre de variables et de contraintes, et l'utilisation de puissantes machines de calcul est nécessaire pour résoudre ceux-ci. Les contraintes et les valeurs connues de ces problèmes sont

malheureusement souvent approximées lors de la modélisation, et les humains ont généralement peu confiance dans les solutions sorties de boîtes noires.

Les systèmes à initiative partagée offrent un cadre de collaboration entre différents acteurs où chacun met à profit ses forces pour accomplir une tâche. Dans le cas de l'optimisation d'un réseau de création de valeur, il est possible de faire intervenir un ou des décideurs humains pour compenser l'imprécision des données par leur intuition, peaufiner la recherche de solution optimale par leur expertise et augmenter la valeur perçue de la solution obtenue par leur intervention.

Hamel *et al.* utilisent les algorithmes de résolution de problèmes d'optimisation linéaire et les propriétés de convexité des espaces solutions de ceux-ci pour proposer un système à initiative partagée pour l'optimisation d'un réseau de création de valeur. Ce système fonctionne, mais il présente cependant des limites au niveau de la présentation des données, de la couverture de l'espace solutions et de la souplesse face à la valeur optimale calculée.

L'outil Logilab offre une présentation de données efficace pour l'affichage d'un modèle linéaire dans le contexte d'un réseau de création de valeur dans l'industrie forestière. Il est possible de l'utiliser comme base pour améliorer la présentation des données et supporter l'interaction proposée par Hamel *et al.*

En ce qui concerne l'application rigide de l'optimalité calculée par l'ordinateur du modèle et de l'augmentation de la couverture de l'espace solutions, plus de recherches doivent être faites. Les différentes voies explorées doivent cependant tenir compte des réalités de l'informatique contemporaine, notamment des limites imposées par l'instabilité numérique des processeurs binaires.

Le chapitre suivant propose une interface inspirée de Logilab pour présenter les données de façon plus claires et conviviales à l'acteur humain au sein du SIP. Le chapitre 3 présente une approche pour gérer l'introduction d'une tolérance à la quasi-optimalité, soit l'inclusion d'une partie de l'ensemble des solutions dont la valeur retournée par la fonction-objectif n'est pas optimale mais qui pourraient

présenter un intérêt pour l'expert humain. Les chapitres 4 et 5 proposent des méthodes pour rendre la totalité de l'ensemble des solutions optimales à l'utilisateur. Le chapitre 4 présente des modifications à l'algorithme du simplexe et le chapitre 5 propose une approche par cartographie de l'espace solutions.

2. Proposition d'interface pour un système à initiative partagée pour l'optimisation d'un réseau de création de valeur

Hamel *et al.* proposent une méthode en deux temps pour l'optimisation interactive d'un réseau de création de valeur au sein d'un système à initiative partagée. Dans un premier temps, un solveur linéaire classique, par exemple l'algorithme du simplexe, détermine la valeur-objectif optimale pour le modèle. Puis, en introduisant une contrainte obligeant la fonction-objectif à prendre cette valeur dans le modèle, il produit un ensemble de solutions optimales indexées en fonction des variables dont les valeurs peuvent être manipulées par l'utilisateur humain. Par la suite, l'utilisateur peut modifier les valeurs de certaines variables de la solution, et le système calcule une combinaison convexe entre les solutions connues pour produire une nouvelle solution optimale prenant en compte les entrées du décideur humain.

L'interface proposée par Hamel *et al.* offre un affichage clair de la zone d'optimalité de chacune des variables modifiables par le décideur. Elle présente cependant les données de façon brute, laissant à la charge de l'utilisateur la mise en contexte de celles-ci. Un meilleur modèle de présentation de données permet à l'utilisateur de comprendre l'impact réel des valeurs assignées aux variables plus facilement et plus rapidement, ce qui lui permet de concentrer son expertise sur la tâche à réaliser plutôt que l'analyse des données.

Le présent chapitre du mémoire présente le développement d'une proposition d'interface pour un système à initiative partagée pour l'optimisation d'un réseau de création de valeur. La première sous-section traite de l'identification de l'utilisateur type du système développé. L'analyse des besoins fonctionnels de l'interface est ensuite présentée. Suit la description du processus itératif de conception des modèles de présentation et d'interaction de l'interface. Finalement, les particularités de l'implémentation de l'interface sont présentées.

2.1. Identification et caractérisation des utilisateurs

La première étape de la conception d'une interface est l'identification et la caractérisation des utilisateurs de ladite interface. N'ayant pas pu rencontrer tous les utilisateurs potentiels directement, le processus fut réalisé avec l'aide de M. Jonathan Gaudreault, directeur de recherche du présent travail, lui-même utilisateur potentiel du système et en contact avec les autres utilisateurs potentiels. Deux utilisateurs types ont été identifiés, soit le chercheur et le partenaire industriel.

Le chercheur est un étudiant ou un professionnel de recherche qui développe de nouvelles méthodes de modélisation et de planification des opérations pour l'industrie forestière. Informaticien, ingénieur ou scientifique, son objectif est d'abord et avant tout d'obtenir rapidement une quantification de la valeur de ses hypothèses. Habitué aux outils de simulation informatique, il est prêt à investir du temps pour apprendre à utiliser un outil moins convivial si celui-ci est efficace.

Le partenaire industriel est un gestionnaire d'entreprise d'envergure provinciale à internationale liée à l'exploitation et/ou à la transformation des produits du bois. Il veut optimiser le rendement de son entreprise, et l'optimisation du réseau de création de valeur de celle-ci est un moyen d'y arriver. Il a une grande expérience en gestion des réseaux de création de valeur et en planification tactique des opérations, mais n'est pas nécessairement habitué aux outils informatiques. Il lui faut donc une solution facile à utiliser et qui affiche clairement l'information.

Le chercheur étant habitué aux systèmes informatiques existants, il a peu d'exigences pour l'interface graphique. C'est donc le partenaire industriel qui sera l'utilisateur type pour la conception de l'interface. Le détail des expertises du partenaire industriel est présenté dans le tableau 1.

Tableau 1 - Expertises de l'utilisateur type

Nature	Niveau	Fréquence d'utilisation
Informatique	Intermédiaire	À chaque exercice de planification
réseau de création de valeur	Expert	À chaque exercice de planification
Planification	Expert	À chaque exercice de planification

L'utilisateur type a trois types d'objectifs lors de l'utilisation du système, soit des objectifs personnels, des objectifs d'expérience et des objectifs finaux. Ces objectifs personnels sont de rendre son entreprise non seulement rentable mais surtout compétitive. Pour ce faire, il désire obtenir rapidement de l'information pour pouvoir prendre des décisions de façon réactive, par exemple lors de réunions de gestion. Lors de son expérience de l'utilisation du système, il cherche à se sentir productif et en contrôle du processus. Il désire à terme obtenir un plan tactique d'opération complet et optimal pour son entreprise, soit résoudre un PCVP.

Il est à noter que le partenaire industriel a déjà investi dans le système LogiLab. La solution proposée doit donc y ressembler, d'une part parce qu'il connaît ce système et, d'autre part, pour valider la pertinence de ses investissements dans les projets de recherche.

2.2. Définition des processus de travail et des modèles de tâches

Le processus de travail devant être supporté par l'interface correspond au processus d'optimisation linéaire interactif proposé par Hamel *et al.* [1] et présenté dans les notions préliminaires du présent ouvrage. Plus précisément, nous nous intéressons à l'interaction en temps réel entre le décideur et le système informatique. L'information de la solution courante est présentée au décideur qui peut alors ajuster interactivement les valeurs de certaines variables du modèle jusqu'à ce qu'il obtienne une solution qui lui est acceptable. Ce processus est présenté à la figure 7.

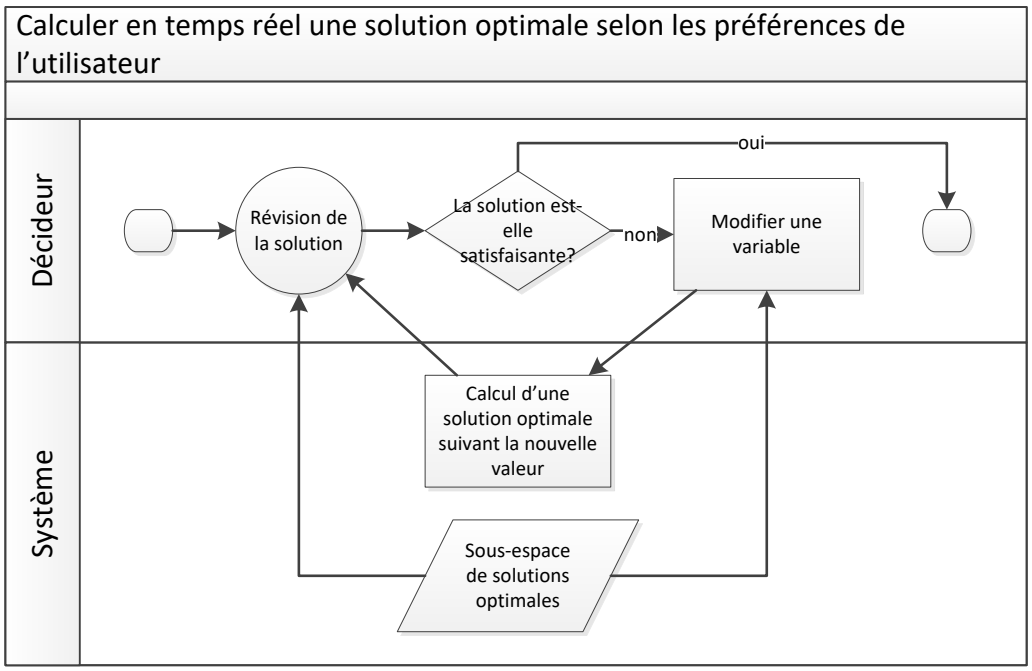

Figure 7- Processus interactif d'optimisation

Ce processus correspond au cas d'utilisation suivant, dont l'objectif est d'ajuster les valeurs des éléments du plan tactique d'opération :

1- Le système présente l'ensemble des variables du réseau de création de valeur à l'utilisateur. Pour chacune d'elles, le système affiche la valeur de la solution suggérée et les valeurs limites de celle-ci dans le sous-espace de solutions optimales.

2- L'utilisateur choisit une variable et lui assigne une nouvelle valeur dans la zone d'optimalité.

3- Le système génère et présente une nouvelle solution où la variable a la valeur fixée par l'utilisateur.

4- Les étapes 2 et 3 sont répétées jusqu'à ce que l'utilisateur soit satisfait de l'ensemble de la solution.

2.3. Création du modèle de présentation

Un travail de conception de modèle de présentation des valeurs des variables d'un réseau de création de valeur dans l'industrie forestière a déjà été effectué lors du développement de l'outil Logilab [14]. De plus, tel que mentionné dans la section

2.1. Identification et caractérisation des utilisateurs, le partenaire industriel est familier avec le système LogiLab et a déjà investi dans celui-ci. Il s'agit donc d'un bon point de départ pour présenter les données, auquel il faut ajouter les propositions faites par Hamel et *al.*

À l'information du plan collaboratif des ventes et de la production générée par le solveur sont ajoutées l'information de l'espace d'optimalité obtenus par la méthode proposée dans les travaux précédents [1]. Chaque arc et chaque nœud présente donc une valeur actuelle, une valeur minimale et une valeur maximale.

Les activités possibles pour l'utilisateur doivent être les suivantes :

- modifier la valeur actuelle d'une quantité de ressources transportées (arcs);
- modifier la valeur actuelle d'un taux d'utilisation d'un chemin (arcs);
- modifier la valeur actuelle d'un taux d'utilisation de centre d'affaire (nœuds);
- verrouiller une variable (arcs et nœuds).

La conception du modèle de présentation des données permettant d'accomplir les activités requises pour l'accomplissement de la tâche d'optimisation s'est faite de façon itérative. Chaque bloc de données (arcs et nœuds) a été traité séparément. Pour chaque itération, les avantages et les inconvénients de la présentation sont listés.

Version 1 : représentation des valeurs minimales et maximales sur les arcs

La première idée est de superposer les valeurs minimale, maximale et actuelle des quantités de ressources transportées avec des opacités différentes. La valeur minimale est pleinement opaque au milieu, la valeur maximale plus pâle et la valeur actuelle dans la solution présentée est semi-opaque. Le concept est présenté à la figure 8.

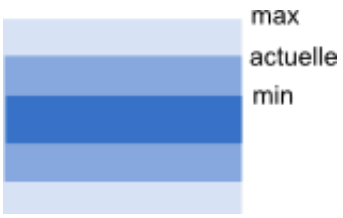

Figure 8- Première proposition de présentation des arcs

Cette présentation a l'avantage d'être simple et directe. Elle met cependant l'emphase sur la valeur minimale plutôt que sur la valeur manipulable par l'utilisateur. De plus, sur certains moniteurs la couleur la plus pâle est difficile à discerner. Finalement, ce modèle ne présente qu'une partie des données.

Version 2 : Emphase sur la valeur actuelle

Afin de déplacer l'emphase visuelle sur la valeur actuelle manipulable, l'écart d'opacité entre les deux valeurs est réduit. De plus, une bordure opaque est ajoutée. La valeur actuelle a une bordure plus foncée afin de la mettre en évidence. Le résultat est présenté à la figure 9.

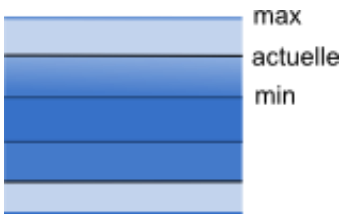

Figure 9 - Seconde proposition de présentation des arcs

Cette présentation est plus claire que la précédente. Elle demeure cependant chargée pour de petites valeurs et ne présente toujours pas le taux d'utilisation du chemin.

Version 3 : Ajout du taux d'utilisation des arcs

Pour la présentation du taux d'utilisation d'un chemin, soit la portion de la capacité théorique de celui-ci, la métaphore d'un tuyau transportant un liquide est utilisée, le

taux d'utilisation correspondant au remplissage vertical de l'arc. Ce tuyau est placé au centre de l'arc.

La couleur d'un taux d'utilisation est interpolée en fonction de sa valeur, le rouge pour un taux de zéro (0) et le vert pour un taux d'un (1). La valeur minimale prend la couleur de la valeur actuelle pour garder l'emphase visuelle sur celle-ci. Les valeurs minimale, maximale et actuelle prennent une opacité différente comme les quantités de ressources transportées.

Il existe deux cas d'exception, soit des chemins ayant une capacité temporairement nulle (e.g. une route impraticable en hiver) ou ayant une capacité non définie. Dans le cas d'un chemin de capacité nulle, il est représenté par un tuyau entièrement rouge. Pour le chemin de capacité indéterminé, on fixe la valeur minimale à 0.25, la valeur actuelle à 0.5 et la valeur maximale à 0.75, les trois de couleur verte. Ces modèles sont présentés à la figure 10.

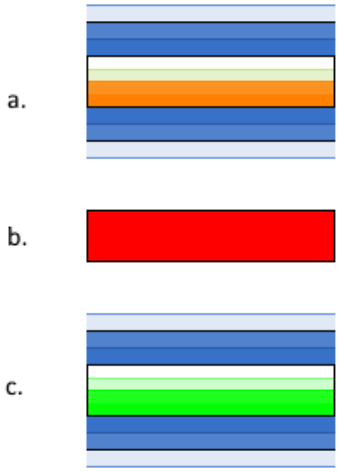

Figure 10 - Ajout des taux d'utilisation des chemins pour a. un chemin normal, b. un chemin à capacité nulle et c. un chemin à capacité indéfinie.

Cette présentation offre toute l'information nécessaire sur les variables modifiables par l'utilisateur. Il existe cependant un risque que ces valeurs soient difficiles à lire pour de petites valeurs considérant que la présentation est chargée en termes de quantité d'information et que celle-ci est concentrée sur de petites régions de l'affichage.

Version 4 : Intégration des arcs dans le graphe

En plus des valeurs minimale, maximale et actuelle des variables de transport de ressources, certains renseignements sur la nature du transport sont affichés par LogiLab en plaçant les arcs de transport de façon à relier les unités d'affaires entre lesquelles les ressources sont transportées, en ajustant la largeur des traits en fonction de leurs valeurs relatives aux autres transports d'un même type de ressources, les transports d'un même type ayant la même couleur, chaque type ayant une couleur propre.

La taille des quantités de ressources transportées par un chemin est fixée en fonction de la valeur maximale de l'espace d'optimalité que peut prendre la variable. La largeur des taux d'utilisation est quant à elle fixée en fonction de la capacité totale de chemins transportant la même unité de mesure à l'exception des chemins de capacité infinie. Tous les chemins ont une largeur d'au moins trois points. Les chemins de capacité infinie ont une largeur d'un point plus grand que le chemin le plus large. Si tous les chemins ont une capacité infinie ou nulle, la largeur est fixée comme étant la moitié de la valeur moyenne des valeurs maximales du taux d'utilisation globale de chaque flèche. La figure 11 présente un exemple de résultats.

Figure 11 - Intégration des arcs dans le graphe

Cette présentation expose toute l'information simultanément, mais devient surchargée et difficile à comprendre, surtout avec la taille plus fine des chemins ayant des valeurs de transport peu élevées.

Version 5 : Épuration de la présentation

Il a été convenu de simplifier en retirant le taux d'utilisation de chaque chemin de l'affichage principal et d'offrir en option d'ajuster les affichages en fonction des capacités.

La présentation se fera par « tuyau » dont la largeur représente la valeur maximale en fonction de l'espace d'optimisation. De façon similaire à la présentation de l'itération 2, la valeur minimale d'optimalité est représentée par une barre de pleine opacité bordée d'une ligne un ton plus foncé, la valeur maximale est représentée par une barre de basse opacité marquée d'une ligne un ton plus clair et la valeur courante est représentée par une barre d'opacité moyenne bordée de noir. Cependant, les barres sont alignées au bas, de façon à donner plus d'espace entre les valeurs et à renforcer la métaphore du tuyau.

Les largeurs des tuyaux pour l'affichage principal sont fixées comme suit : celui qui a la plus grande valeur maximale prend la largeur maximale autorisée pour un tuyau et les autres ont une largeur proportionnelle à celui-ci. Le code de couleur en fonction de la nature de la ressource transportée de Logilab est conservé. Ce concept est présenté à la figure 12.

Figure 12 - Épuration de la présentation

Version 6 : Ajout de l'interactivité

Pour permettre à l'utilisateur d'ajuster la valeur d'une quantité de ressources transportées, la façon la plus simple est de simplement cliquer sur la ligne noire et

de la glisser dans la zone affichée (le mouvement est contraint entre les valeurs d'optimalité pour prévenir l'entrée d'une valeur non optimale).

Afin de conserver la cohérence d'information tandis que les variables sont modifiées par l'utilisateur, l'échelle de largeur des tuyaux est fixée à partir des valeurs maximales des espaces d'optimalité et cette échelle est maintenue tout au long de la session de travail.

Cette méthode est simple et intuitive, mais offre peu de précision pour l'ajustement des valeurs.

Version 7 : Amélioration de la précision

Afin de permettre une plus grande précision dans l'ajustement des variables, la zone d'interaction entre le pointeur et l'arc est agrandie lorsque l'utilisateur passe son curseur sur un arc. Cette interaction est présentée à la figure 13.

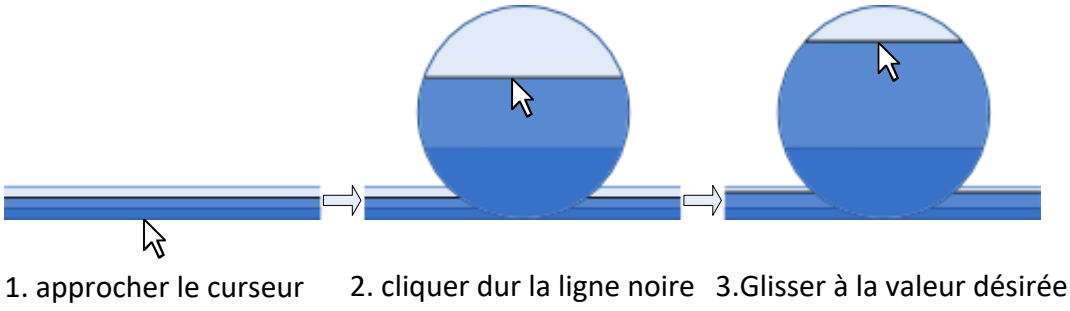

Figure 13 - Agrandissement d'un arc pour l'interactivité

Chaque arc affiche aussi au-dessous de lui l'information sur les valeurs sous forme chiffrée afin de donner les valeurs exactes présentées sous forme $valeur\_min < valeur\_courante < valeur\_max$. Si la valeur courante est égale à l'une des extrémités, on fait alors disparaître l'extrémité en question pour montrer que la borne est atteinte.

Version 8 : ajout des unités d'affaires

Le taux d'utilisation des unités d'affaires est présenté par une simple barre horizontale de pourcentage. On fixe la largeur à la capacité maximale de l'unité fixée par les contraintes. Dans ce champ, on place une barre à triple opacité qui représente horizontalement l'espace d'optimalité et la valeur courante de la variable de façon similaire aux tuyaux, mais avec un code de couleur différent Plutôt que de représenter l'unité de mesure de la variable, la couleur représente le taux d'utilisation de l'unité d'affaires. Elle est interpolée entre vert pour un taux d'utilisation de 1, orange pour un taux de 0,5 et rouge pour un taux de 0. La barre du maximum pour la variable dans l'espace d'optimalité prend la couleur correspondant à son taux, tandis que la barre de la valeur minimale prend la couleur du taux de la valeur courante.

Contrairement aux arcs, la largeur des glissières des unités d'affaire est suffisante pour qu'on puisse interagir directement avec celles-ci. De plus, la représentation des unités d'affaires comporte une icône représentant le type d'unité d'affaires au-dessus de la glissière ainsi que son nom au-dessous. Le tout est présenté à la figure 14.

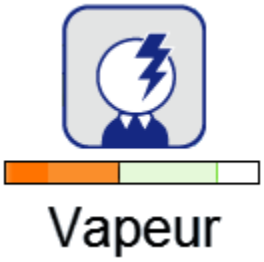

Figure 14 - Présentation d'une unité d'affaire

Version 9 : Présentation de l'objectif global

Afin d'offrir plus de latitude à l'utilisateur dans l'assignation des valeurs aux différentes variables du modèle, le système offre une certaine tolérance à l'optimalité de la variable optimisée. Puisque l'utilisateur cherche à optimiser cet aspect (soit maximiser le profit ou réduire les dépenses), l'impact sur l'optimalité découlant des modifications faites aux variables doit lui être présenté. Pour ce faire, un panneau expose le nom de la variable à optimiser, la valeur que celle-ci prend selon la solution courante et le taux d'optimalité de cette valeur. Pour ajouter à l'impact visuel, une barre de couleur est placée sous ce texte. La couleur est interpolée de manière similaire aux glissières des centres d'affaires, à la différence que le vert est assigné à la valeur optimale, le rouge à la valeur à la limite de la zone de tolérance. Ce panneau est présenté à la figure 15.

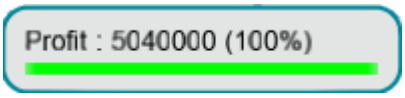

Figure 15 - Présentation de l'objectif global

Version 10 : Navigation dans le temps

Les données affichées par les systèmes sont agrégées, notamment en fonction du temps. Une solution globale est présentée, mais chaque variable de cette solution est l'agrégation des variables pour chacune des périodes de temps du modèle. Afin de donner une bonne information à l'utilisateur, le système permet d'afficher soit le réseau de création de valeur pour l'ensemble des périodes, soit pour une période précise.

Pour permettre une navigation efficace dans les périodes, le système comporte un panneau de navigation. Celui-ci présente la période sélectionnée sous forme alphanumérique (un chiffre pour une période ou « Global » pour la solution agrégée). Cette valeur est placée dans un champ de saisie qui permet à l'utilisateur d'entrer

manuellement une valeur de période arbitraire (si la valeur ne correspond pas à une période existante, le système ignore l'entrée). À côté du champ de saisie se trouve un bouton de flèche gauche et un bouton de flèche droit permettant de passer à la période précédente ou suivante. Puisqu'il s'agit d'une agrégation de l'ensemble des périodes, la période globale est placée à la fin des périodes individuelles. Sous ces éléments, on retrouve une glissière qui montre la position de la période dans le fil du temps. Encore une fois, la période globale étant une agrégation des périodes, elle est placée à la fin de la ligne temporelle, donc à l'extrémité droite de la glissière. Ce design est présenté dans la figure 16.

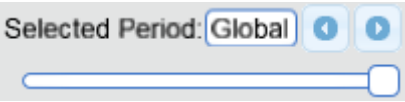

Figure 16 - Navigation dans le temps

L'utilisation d'un des éléments pour changer la période affichée met automatiquement à jour les autres : le champ de saisie affiche la valeur courante, la glissière se positionne et si on se trouve sur l'une des extrémités de la ligne de temps, le bouton permettant un déplacement dans la direction de celle-ci est désactivé.

Version 11 : Navigation dans les modifications

Le but du système étant de permettre l'exploration de l'espace solutions, il est probable que l'utilisateur fera plusieurs modifications qu'il ne souhaitera pas conserver. Afin de réduire sa charge mentale et de lui permettre de revenir à des solutions précédentes sans devoir les refaire lui-même de mémoire, le système conserve un historique des dernières modifications avec des fonctions pour annuler et refaire des modifications. Chaque modification de variable est historiée et porte un nom de la variable modifiée afin de permettre à l'utilisateur de savoir à quelle modification correspond chaque élément de l'historique. Le système permet aussi de retourner directement à un point de l'historique.

De plus, l'utilisateur peut enregistrer une solution qui lui plaît pout y revenir même s'il a fait plus de modifications que ne peut en contenir l'historique. Charger une solution est historié sous le nom de la solution chargée avec indication de chargement. La solution suggérée initialement par le solveur est sauvegardée par défaut. Chaque sauvegarde porte comme nom le moment où elle a été enregistrée, à l'exception de la solution initiale.

La dernière modification est affichée dans un champ de menu déroulant dans le panneau de navigation. Une flèche vers le bas indique à l'utilisateur qu'il peut faire dérouler le menu en cliquant sur la dernière modification. Le menu déroulant présente les modifications les plus récentes ordonnées avec la plus récente au-dessus. Chaque modification porte aussi un numéro représentant sa position dans le temps par rapport à la modification la plus récente dans l'historique. En cliquant sur une modification de l'historique, l'utilisateur peut aller directement à celle-ci. Les modifications ultérieures restent disponibles tant qu'aucun autre événement n'est historié.

À droite de ce champ se trouve deux boutons, un à gauche et un à droite, qui permettent respectivement d'annuler ou refaire une modification. Encore une fois, lors d'une annulation, les modifications ultérieures restent disponibles tant qu'aucun autre événement n'est historié. Peu importe le moyen choisi pour faire un déplacement dans l'historique des modifications, tous les éléments sont mis à jour : le champ de menu déroulant affiche la dernière modification appliquée à la solution affichée, la liste de l'historique contient les points d'historique accessibles et si on se trouve à l'une des extrémités de l'historique, le bouton permettant de se déplacer vers celle-ci direction et désactivé. Ces éléments sont présentés à la figure 17.

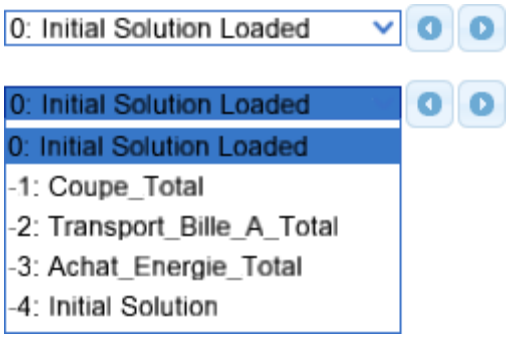

Figure 17 - Navigation dans l'historique des modifications

Sous les éléments de navigation dans l'historique se trouve un champ de liste déroulante comportant la mention « Saved Solutions ». Cette liste contient les solutions enregistrées. Une flèche vers le bas indique à l'utilisateur qu'il peut faire dérouler le menu en cliquant sur le champ. En cliquant sur une solution de la liste, celle-ci est chargée et affichée. Cette opération est historiée, et donc le chargement est affiché dans le champ de l'historique des modifications. À droite du champ se trouve un bouton avec l'icône d'une disquette. En cliquant sur ce bouton, la solution courante est enregistrée et ajoutée à la liste. Ces éléments sont illustrés à la figure 18.

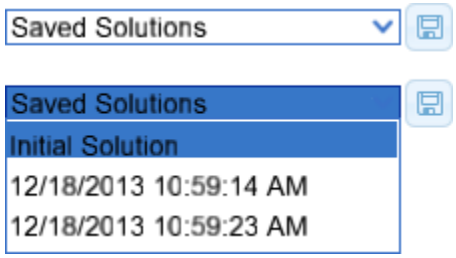

Figure 18 - Navigation dans les solutions sauvegardées

La navigation dans le temps, les modifications et les solutions sauvegardées sont rassemblées dans le panneau de navigation. Ce panneau est présenté à la figure 19.

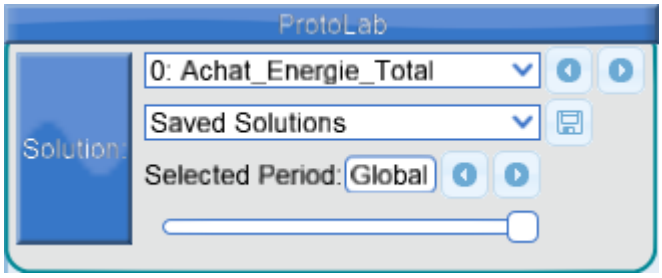

Figure 19- Panneau de navigation

Version 12 : Impact sur les variables non affichées

Bien que l'interface n'affiche qu'une partie des variables, soit l'agrégation pour l'ensemble des périodes ou les données pour un période donnée, chaque modification faite par l'utilisateur affecte potentiellement toutes les variables du modèle. Il serait trop lourd d'afficher toutes les variables (un modèle type comporte 52 périodes d'une semaine pour un plan annuel).

Afin de fournir à l'utilisateur une indication de l'impact des modifications sur le reste du plan, on ajoute sous la barre de navigation une série de colonnes indiquant la variation normalisée moyenne des variables d'une période. Pour calculer cette valeur, on mesure la variation de chaque variable par rapport à la grandeur de sa zone d'optimalité et on fait la moyenne des variables du groupe. La hauteur de la colonne augmente avec la valeur de variation et sa couleur est interpolée entre le vert et le rouge afin d'attirer l'attention sur les périodes où la variation est la plus forte.

Chaque colonne est située vis-à-vis de la période concernée. Afin de faciliter la lecture de l'information, la barre de navigation dans les périodes est retirée du panneau de navigation et envoyée dans un nouveau panneau au bas de l'interface avec les colonnes de visualisation des impacts. L'ensemble des éléments est présenté à la figure 20.

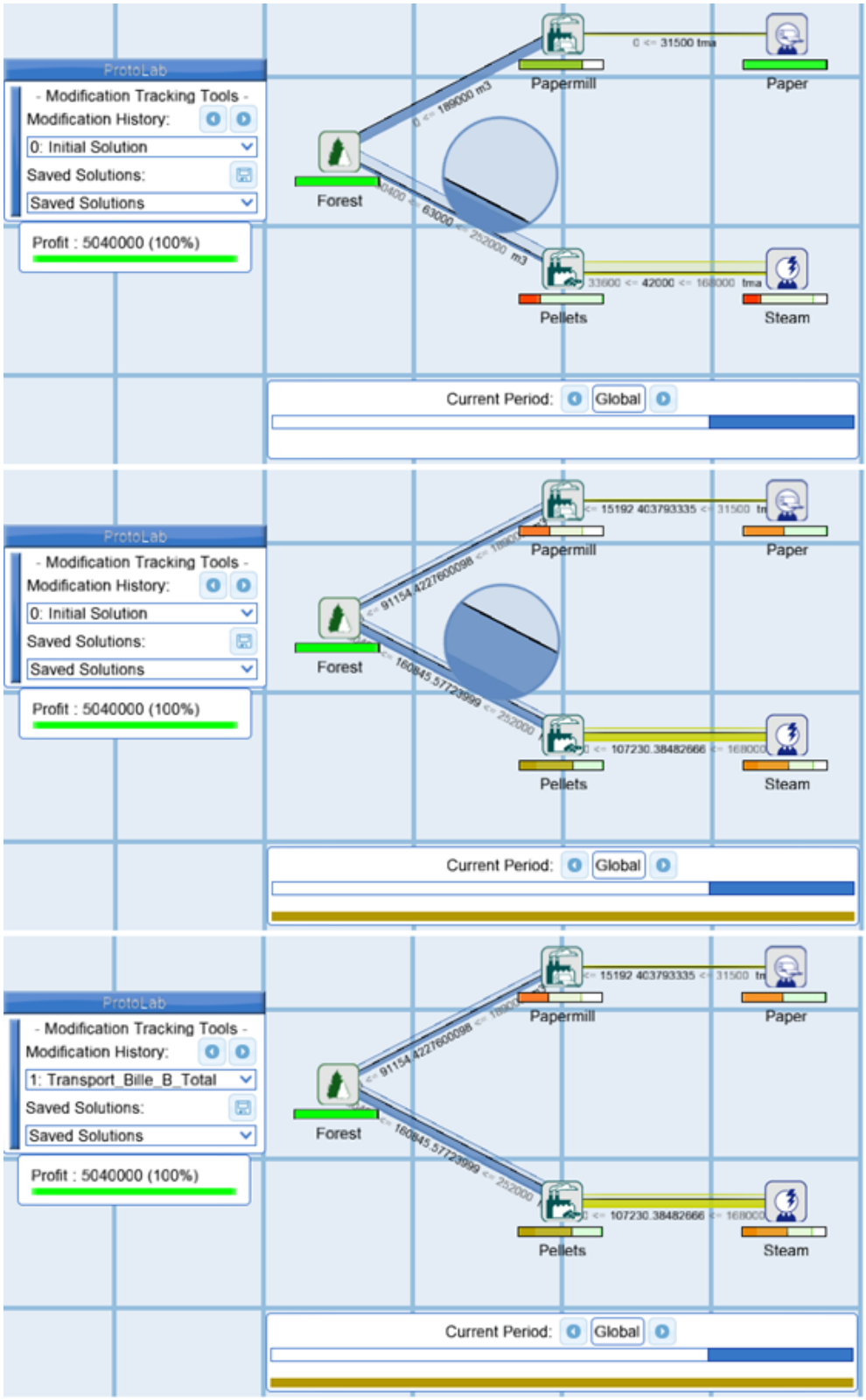

Figure 20 - Présentation complète des éléments de l'interface

2.4. Proposition de deux approches pour la récupération des solutions obtenues précédemment.

La génération interactive de solution selon l'heuristique triangulaire proposée par Hamel *et al.* utilise la combinaison de la solution actuelle et d'une solution extrême pour générer une nouvelle solution. Réassigner la valeur précédente à une variable fera une combinaison de cette nouvelle solution et de la solution extrême associée à l'autre extremum de cette variable. À moins que la solution actuelle initiale et première solution générée ne soient les solutions extrêmes liées à la variable dont la valeur est modifiée, les solutions combinées ne sont pas les mêmes et, par conséquent, la manipulation ne produira pas un retour à la solution précédente. Ce scénario est présenté à la figure 21.

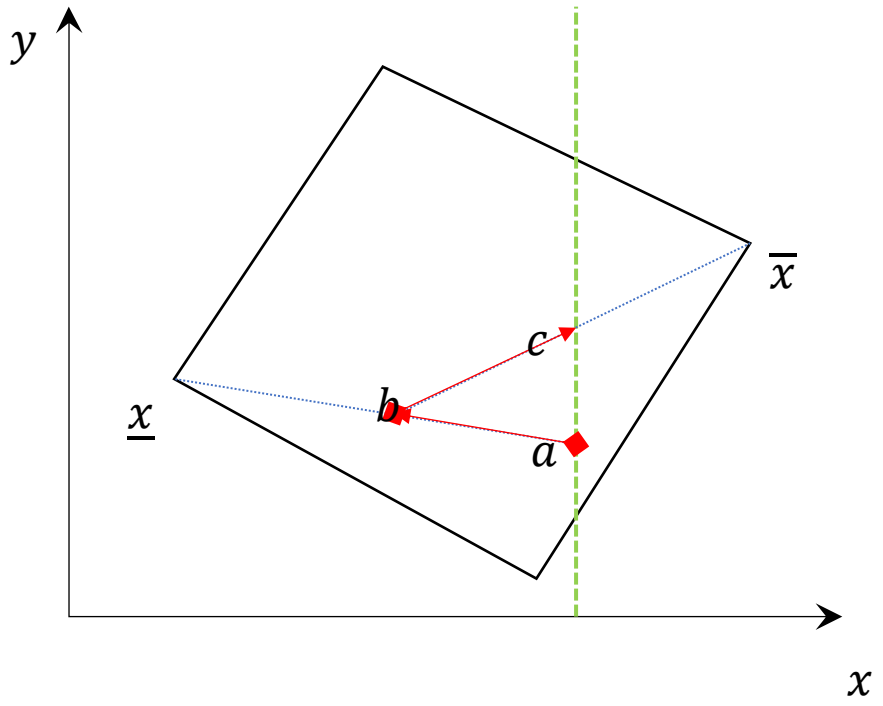

Figure 21 – Perte de la solution précente pour une manipulation inverse. La solution est originalement en a. L'utilisateur diminue la valeur de la variable x, ce qui produit la solution b. Redonner la valeur préceédente à la variable x retourne la solution c et non la solution précédente a.

Pour permettre à l'utilisateur de tester une modification sans risquer de ne plus pouvoir revenir à la solution actuelle si la nouvelle solution n'est pas aussi bonne selon son jugement, il faut prévoir la mémorisation des solutions précédentes et une méthode de retour à celles-ci.

Deux approches pour gérer les solutions enregistrées et leur récupération ont été envisagées : le journal de modifications et l'historique simple. Le journal de modifications consiste à mémoriser la succession des combinaisons convexes liées aux modifications faites par l'utilisateur, tandis que l'historique simple conserve la totalité de chaque solution générée. Les opérations que les deux méthodes doivent supporter sont:

- obtenir une nouvelle solution;
- effectuer un changement de période;
- revenir à une modification précédente.

Les critères pour un bon système à initiative partagée sont le temps de réaction et la stabilité lors des manipulations par l'utilisateur [1]. La stabilité n'est cependant pas applicable au passage à une modification précédente puisqu'il s'agit d'un saut. La meilleure solution pour le retour à une modification précédente est donc celle demandant le plus petit temps de calcul, c'est-à-dire la moins grande complexité algorithmique. Ce critère est surtout important pour l'obtention d'une nouvelle solution, l'opération la plus courante.

Pour évaluer chaque approche en termes de complexité algorithmique, posons :

- p, le nombre de périodes;
- n, le nombre de variables;
- m, le nombre de modifications effectuées par l'utilisateur à la solution en mémoire;
- S, un tableau de réels $\mathbb{R}^{2n+1\times n}$ contenant l'espace d'optimalité selon le modèle proposé par Hamel *et al.* [1];

- $Combiner(\mathbb{R}^n, \mathbb{R}^n, \alpha)$, une opération de combinaison convexe de complexité temporelle d'ordre $O(n)$;
- J, une liste de modifications encodées par des tuples $\langle \mathbb{N}^+, \mathbb{R} \rangle$ (l'entier naturel étant l'index de la solution à interpoler avec la courante, et le réel étant la valeur du taux d'interpolation);
- H, une liste circulaire de $\mathbb{R}^n$ contenant un historique des solutions.

Pour les deux approches, lors d'une combinaison, il suffit de faire appel à la fonction $Combiner$, soit une opération d'ordre $O(n)$. De même, lors d'un changement de période, il suffit de mettre à jour les variables assignées aux éléments d'affichage, une opération d'ordre $O\left(\frac{n}{p}\right)$.

Pour effectuer le retour à une modification précédente, le journal de modifications et l'historique procèdent différemment. Le journal de modifications doit refaire la séquence de modifications menant jusqu'à la solution désirée, soit une opération en $O(mn)$. L'historique simple se contente pour cette tâche de récupérer la solution dans le tableau, soit une opération d'ordre $O(1)$.

Il est possible d'optimiser la combinaison sur le journal de modifications en affectant uniquement les valeurs affichées. L'opération passe alors à un ordre de $O\left(\frac{n}{p}\right)$, et le retour à une solution précendente devient d'ordre $O\left(\frac{mn}{p}\right)$. En contrepartie, le changement de période implique de recalculer la solution courante comme un retour à une solution précédente, pour une opération d'ordre $O\left(\frac{mn}{p}\right)$. De plus, cette optimisation implique que l'on ne calcule pas l'impact d'une modification sur les variables non affichées dans le graphe.

La complexité algorithmique de chacune des approches pour chacune des tâches est résumée au tableau 2.

Tableau 2 - Comparaison de la complexité algorithmique des différentes approches envisagées pour l'implémentation du retour aux modifications précédentes.

Erreur ! Source du renvoi introuvable.**Tâche**	**Approche**			
	Historique	Journal		
	simple	simple	périodisé	limité
Obtenir une nouvelle solution	$O(n)$	$O(n)$	$O\left(\frac{n}{p}\right)$	$O\left(n+\frac{n}{p}\right)$
Changer de période	$O(1)$	$O(1)$	$O\left(\frac{mn}{p}\right)$	$O\left(\frac{ln}{p}\right)$
Revenir à une période précédente	$O(1)$	$O(mn)$	$O\left(\frac{mn}{p}\right)$	$O\left(\frac{ln}{p}\right)$

Le journal périodisé offre la meilleure performance avec sa complexité algorithmique de $O\left(\frac{n}{p}\right)$, mais il ne permet pas de connaître l'impact des modifications effectuées sur les valeurs des variables non affichées. L'historique simple et le journal de modifications simple sont *ex aequo* en terme de complexité algorithmique pour l'obtention d'un nouvelle solution et le changement de période, mais l'historique simple est plus rapide pour le retour à une période précédente. Il s'agit donc du meilleur choix.

Le contre-coût de l'historique simple est l'espace mémoire à utiliser, mais en considérant un espace solutions de $(2n+1)n$, où n est généralement de l'ordre de 10^5 ou 10^6 dans les cas industriels réels, conserver l solutions, où $l < 10^2$, ln représente un coût mémoire minime en comparaison.

En termes de mémoire, le journal de changements enregistre un entier et un réel pour chaque modification. Cependant, puisque la solution source de la combinaison n'est généralement pas une solution enregistrée dans la définition de l'espace d'optimalité, on doit aussi conserver une solution de départ, soit une taille mémoire totale de $2m+n$. L'historique simple quant à lui doit mémoriser la totalité de chacune des m solutions et a donc une taille de mn.

Il est possible de limiter la mémoire utilisée en ne conservant que les l dernières modifications et en utilisant une liste circulaire pour les stocker. Pour l'historique simple, cette méthode réduit directement la taille de la mémoire à lm. Pour le journal de modifications, le coût en mémoire est de $2l + n$ et demande d'effectuer une combinaison supplémentaire à chaque génération d'une nouvelle solution pour maintenir la solution de départ à jour.

Les solutions pour limiter le coût en mémoire de l'historique simple permettent donc de justifier le choix de cette méthode pour gérer le retour aux solutions précédentes. C'est cette méthode qui a été retenue lors de l'implémentation de l'interface pour valider celle-ci.

2.5. Implémentation de l'interface

Afin de valider le design de l'interface auprès des utilisateurs lors de son utilisation avec les algorithmes développés par Hamel *et al.*, une implémentation du système en C# au sein du moteur de jeu Unity a été réalisée. Le module NGUI a été utilisé pour faciliter la réalisation de l'interface graphique.

Le système repose sur trois couches logicielles représentant des niveaux d'abstraction. La première couche correspond au modèle mathématique linéaire sur lequel les méthodes développées par Hamel *et al.* peuvent être appliquées, en faisant abstraction de ce que le modèle représente réellement. La seconde couche est l'organisation des variables en réseau de création de valeur périodisé, qui fait abstraction du modèle linéaire de contrainte et des méthodes moyens d'affichage et de manipulation des solutions. La troisième couche contient les mécanismes de manipulation et d'affichage élaborés dans l'interface humain-machine. Ce découpage facilite la portabilité et la réutilisabilité de la solution développée.

La couche mathématique s'articule autour de l'objet ModelManager. Cet objet contient les variables en objets Variable qui contiennent l'information sur la valeur courante ainsi que les limites d'optimalité et de valeur de celle-ci. Il contient aussi un

objet Triangulator qui s'occupe d'appliquer l'heuristique triangulaire pour générer de nouvelles solutions en temps réel. Il comporte aussi un objet SolverCaller qui fait les appels au serveur ou à la grappe de calcul qui peut générer l'espace d'optimalité. Finallement, le ModelManager contient aussi un ModificationHistory qui gère l'historique des modifications faites par l'utilisateur, conservées dans des objets SavedSolution.

La couche de réseau logistique périodisé est gérée par l'objet PLogisticNetworkManager. Celui-ci transmet les demandes de modification de la solution au modèle mathématique et reçoit la notification de mise à jour du graphe lorsqu'une nouvelle solution est disponible. Le PLogisticNetworkManager contient des objets PLNElement, chacun lié à une variable du modèle mathématique. Il existe trois types de PLNElement, soit les objets Flow, Node et Objective, correspondant respectivement aux flux de ressources et produits (les arcs du graphe), aux unités d'affaires (les noeuds du graphe) et à la fonction-objectif du modèle.

Finalement, la couche de l'interface humain-machine graphique est gérée par l'objet UIPLNDisplayManager. Cet objet contient des objets UIDisplayElements qui affichent les valeurs assignées aux variables de la solution courante ainsi que leur zone d'optimalité. Les UIDisplayElements permettent aussi de récupérer les modifications faites par l'utilisateur sur les valeurs assignées aux variables. Il existe trois sortes d'objet UIDisplayElement, soit les UIPipe, UIUnit et UIBox, qui permettent d'intéragir avec respectivement les valeurs des variables liées aux flux de ressources et produits, des unités d'affaires et la valeur objetif. Le UIPLNDisplayManager contient aussi un objet UIZoomBubble qui permet d'assigner avec plus de précision les valeurs aux variables de flux. Finalement, UINavigator s'occupe de gérer la navigation entre les périodes et dans l'historique de manipulation. Cette architecture est présentée à la figure 22.

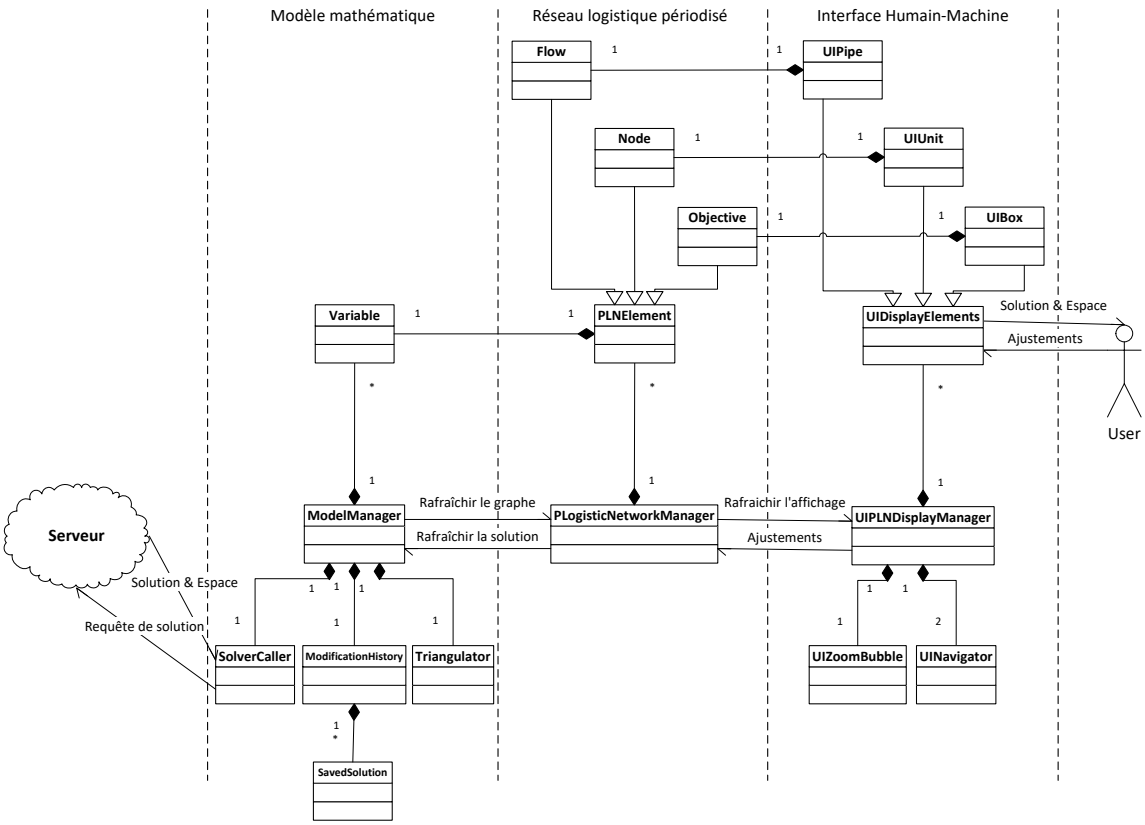

Figure 22 - Diagramme UML de haut niveau de l'architecture du système à initiative partagée proposé

2.6. Conclusion

L'utilisateur cible de l'outil a été identifié comme étant un gestionnaire de réseau de création de valeur dans une entreprise forestière. L'approche interactive proposée par Hamel *et al.* a servi de cas d'utilisation pour l'élaboration de la nouvelle interface proposée. Cette interface est basée sur celle de l'outil Logilab, à laquelle la présentation des nouvelles données utiles au processus interactif ont été ajoutées itérativement.

Une architecture en trois couches logiques a permis une implémentation de l'interface avec les algorithmes de Hamel *et al.* pour valider sa fonctionnalité.

La clarté de la présentation des donnnées permet au gestionnaire de mettre à profit son expérience et son intuition pour diriger la recherche d'une solution au sein de l'espace d'optimalité stricte. Elle ne lui permet cependant pas d'aller explorer des solutions au-delà de cet espace.

3. Approche permettant une tolérance quant à l'optimalité des solutions

La méthode de Hamel limite l'utilisateur à l'exploration d'un espace de solutions optimales. Cette valeur optimale est établie lors de la première résolution du modèle et est ensuite imposée de manière intransigeante, sans tenir compte des aspects qualitatifs mesurés par l'intuition et l'expérience du décideur humain. On peut supposer que des solutions presque optimales pourraient être tout autant acceptables. Qui plus est, la possibilité d'explorer certaines solutions en dehors de l'espace des solutions strictement optimales permettrait une plus grande participation du décideur humain à l'élaboration de la solution finale et d'augmenter la pertinence et l'acceptation de celle-ci.

Dans ce qui suit, nous appellerons *valeur-objectif* d'une solution la mesure obtenue par l'application de la fonction-objectif à cette solution. Par exemple, si on cherche à maximiser la valeur de $x + y$, la valeur-objectif de la solution $x = 3.5, y = 6.5$ est de 10.

Une *solution strictement optimale* se définit comme une solution dont la valeur-objectif est optimale en vertu des contraintes et de l'objectif d'optimisation. Pour la solution précédente, si les contraintes sont $x, y \geq 0$ et $x + y \leq 10$, la solution est *strictement optimale*.

Dans l'exemple précédent, toute solution pour laquelle $x + y < 10$ est sous-optimale. Prenons la solution $x = 3.4, y = 6.4$. Cette solution a une valeur-objectif de 9.8, ce qui diverge de la valeur optimale par 2 %. Cette divergence est relativement petite et pourrait être acceptable pour accommoder des critères d'évaluation qualitatifs estimés par l'utilisateur humain. On parle d'une *solution quasi optimale* si la divergence entre valeur-objectif de celle-ci et sa valeur strictement optimale est inférieure à un ***seuil de tolérance***. La définition d'un seuil de tolérance permet de délimiter un espace de solutions plus large, contenant des solutions quasi

optimales (mais acceptables par l'utilisateur) en plus des solutions optimales. On perdra donc en optimalité pour gagner en agilité.

Ce chapitre traite précisément de cet élargissement de l'espace de solutions par l'introduction d'un seuil de tolérance. Tout d'abord, une approche naïve pour la gestion de la tolérance dans la méthode de Hamel *et al.* est présentée. Les limites de cette approche sont révélées et une approche dite ***multiétape*** est présentée pour pallier ces problèmes. La méthode de Hamel et les deux nouvelles approches sont ensuite appliquées à un problème simple pour comparer de façon quantifiable les performances obtenues par chacune.

3.1. Intégration naïve de la quasi-optimalité à l'approche de Hamel

La façon la plus simple d'intégrer la tolérance à la quasi-optimalité dans l'approche proposée par Hamel est de générer les solutions correspondant aux extrema du domaine des variables modifiables en imposant le seuil de quasi-optimalité toléré plutôt que l'optimalité stricte. Cette approche naïve crée cependant un problème lors de l'exécution de l'heuristique triangulaire.

Avec cette heuristique, la valeur-objectif de la nouvelle solution est générée par une interpolation (se situant entre les valeurs des solutions combinées, au prorata des poids attribués à chacune). Ceci implique que si l'une des deux solutions combinées a une valeur-objectif qui n'est pas optimale, la valeur-objectif de la nouvelle solution générée ne sera pas optimale non plus. Par exemple, si on combine une solution optimale avec une solution optimale à 96 % (avec un ratio de 0.5 chacune), la solution obtenue sera optimale à 98 %. Cela entraîne deux problèmes dans le comportement du système. D'une part, dès que l'utilisateur ajuste une variable dans une direction pour laquelle la solution extrême est sous-optimale, le système générera une solution systématiquement sous-optimale, même si une solution optimale répondant aux mêmes contraintes peut exister. D'autre part, dès que le système a généré une solution non-optimale, toutes les solutions générées ultérieurement seront sous-optimales jusqu'à ce que l'utilisateur pousse une variable

jusqu'à une valeur extrême correspondant à une solution optimale. La figure 23 présente cette situation de manière visuelle.

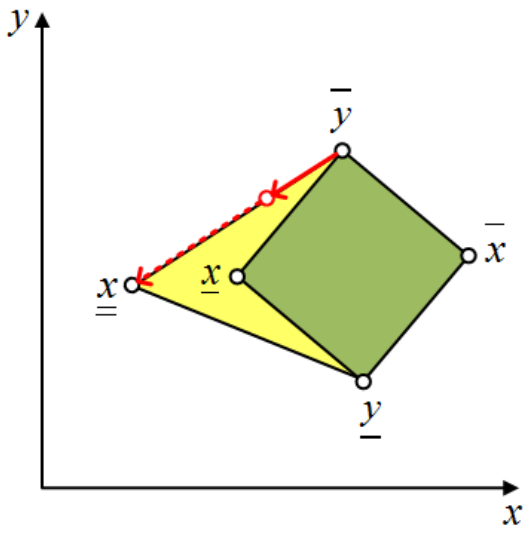

Figure 23 - Comportement de l'heuristique triangulaire de Hamel avec l'introduction de solution quasi optimale. La solution retournée par l'interpolation de $\overline{y}$ et $\underline{\underline{x}}$ retourne une solution quasi optimale alors qu'il existe une des solutions optimales pour lequelle y a la valeur désirée.

En bref, le système ne remplit pas son rôle en ne calculant pas une solution optimale même si c'est possible. Cette défaillance est doublement problématique car elle affecte aussi l'efficacité du décideur humain. En effet, celui-ci se fie au système pour mesurer objectivement la valeur de ses intuitions. En retournant une solution sous-optimale, le système indique au décideur que cette manipulation implique une diminution objective de la valeur, même si ce n'est pas vrai. Le décideur pourrait donc être amené à ignorer des intuitons valides.

Il est plutôt nécessaire que le système retourne une solution optimale lorsqu'il en existe une et ne retourne une solution quasi optimale que s'il n'existe pas de solution optimale pour la valeur entrée par l'utilisateur. Dans ce but, nous proposons une approche dite ***multiétape***.

3.2. Proposition d'une approche multiétape pour intégrer la quasi-optimalité

Comme pour la méthode originale de Hamel, on conserve pour chaque variable modifiable x du modèle, une paire de solutions $\{\underline{x}, \overline{x}\}$, où $\underline{x}$ est une solution optimale minimisant la valeur de x et $\overline{x}$, est une solution optimale maximisant la valeur de x. On ajoute cependant une deuxième paire de solutions $\{\underline{\underline{x}}, \overline{\overline{x}}\}$, $\underline{\underline{x}}$ minimisant la valeur de x et $\overline{\overline{x}}$ maximisant la valeur de x, sous contrainte qu'elle se situe dans une marge d'optimalité définie par le décideur (le ***seuil de tolérance***).

Nous modifions l'heuristique triangulaire pour n'utiliser les solutions $\underline{\underline{x}}$ et $\overline{\overline{x}}$ seulement lorsque la valeur assignée à x par l'utilisateur est en dehors de la zone d'optimalité de cette variable. Par exemple, supposons que la solution courante $c = \overline{y}$ et que l'utilisateur diminue la valeur de x. Tant que la valeur assignée est supérieure à la valeur de x dans $\underline{x}$, on combine entre la solution courante et $\underline{x}$ pour générer la nouvelle solution c'. Dès que la valeur assignée devient inférieure à la valeur de x dans $\underline{x}$, on combine entre $\underline{x}$ et $\underline{\underline{x}}$ pour générer les nouvelles solutions. Cet exemple est illustré à la figure 24.

Formellement, pour une valeur x' assignée à x :

$$c' = \begin{cases} \alpha c + (1-\alpha)\overline{x} \; tel \; que \; \alpha = \dfrac{\overline{x}_x - x'}{\overline{x}_x - c_x}, & \overline{x}_x > x' > c_x \\ \alpha \overline{x} + (1-\alpha)\overline{\overline{x}} \; tel \; que \; \alpha = \dfrac{\overline{\overline{x}}_x - x'}{\overline{\overline{x}}_x - \overline{x}_x}, & \overline{x}_x < x' > c_x \\ \alpha c + (1-\alpha)\underline{x} \; tel \; que \; \alpha = \dfrac{\underline{x}_x - x'}{\underline{x}_x - c_x}, & \underline{x}_x > x' < c_x \\ \alpha \underline{x} + (1-\alpha)\underline{\underline{x}} \; tel \; que \; \alpha = \dfrac{\underline{x}_x - x'}{\underline{\underline{x}}_x - c_x}, & \underline{x}_x < x' < c_x \end{cases} \quad (4)$$

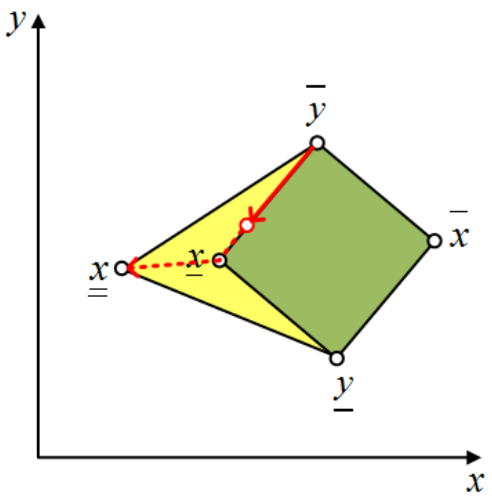

Figure 24 - Exemple de comportement de l'heuristique multiétape

3.3. Cartographie de l'espace de quasi-optimalité

Afin de pouvoir utiliser l'approche multi-étape, il faut générer un espace de solutions explorables défini à la fois par les solutions optimales pour les extrema des valeurs des variables modifiables et les solutions quasi optimales pour ces mêmes extrema. L'approche proposée est similaire à celle de Hamel, soit un tableau de solutions S. Cependant, chaque variable x occupe maintenant quatre lignes au lieu de deux, soit une ligne pour chacune des solutions $\underline{\underline{x}}$, $\underline{x}$, $\overline{x}$, $\overline{\overline{x}}$. Pour une variable indexée à i, la solution $\underline{\underline{i}}$ correspond à $S_{(4i+1)}$, $\underline{i}$ correspond à $S_{(4i+2)}$, $\overline{i}$ correspond à $S_{(4i+3)}$, $\overline{\overline{i}}$ correspond à $S_{(4i+4)}$ (la ligne S_0 étant la solution optimale initiale). Pour un modèle de n variables, on a une taille mémoire de $(4n+1)n$.

Pour générer cet espace de solutions, l'algorithme de Hamel est modifié comme suit :

Soit un modèle linéaire L sur l'ensemble N de n variables, $c(\mathbb{R}^n)$ étant la fonction retournant la valeur d'une solution et o indiquant s'il faut maximiser ou minimiser $c(\mathbb{R}^n)$ et $\epsilon \in [0,1]$ la tolérance relative à l'optimalité.

Algorithme $Relax(L, c(\mathbb{R}^n), o)$

$S \leftarrow \mathbb{R}^{4n+1,n+1}$

$S_0 \leftarrow \begin{matrix} max(c(\mathbb{R}^n), L), & o = max \\ min(c(\mathbb{R}^n), L), & o = min \end{matrix}$

$L' \leftarrow L \cup c(S_0) = c(S_0)$

$L'' \leftarrow \begin{cases} L \cup c \geq (1-\epsilon)c(S_0)\,, & o = max \\ L \cup c \leq (1+\epsilon)c(s_0), & o = min \end{cases}$

POUR $i \in [0, n]$ FAIRE

$S_{4i+1} \leftarrow min(c(\mathbb{R}^n), L'')$

$S_{4i+2} \leftarrow min(c(\mathbb{R}^n), L')$

$S_{4i+3} \leftarrow max(c(\mathbb{R}^n), L')$

$S_{4i+4} \leftarrow max(c(\mathbb{R}^n), L'')$

3.4. Protocole expérimental

Afin de valider cette approche, nous avons comparé les espaces de solutions explorables générés par la méthode de Hamel (espace solutions strictement optimal et heuristique triangulaire) avec tolérance naïve présentée à la section 3.1 (espace solutions quasi optimal et heuristique triangulaire) et la méthode multiétape présentée à la section 3.2 (espace solutions quasi optimal et heuristique multi étape) sur un modèle simple. Pour chaque méthode, la grandeur du domaine, de la zone d'optimalité et de la zone assignable de chaque variable dans l'espace solutions et l'optimalité des solution générées sont mesurées formellement.

Nous définirons la ***zone assignable*** d'une variable (Δ) comme étant la différence entre la valeur maximale et la valeur minimale d'une variable dans les solutions bornant l'espace de solution explorables. Pour les méthodes de Hamel, la zone assignable correspond exactement à la zone d'optimalité, définie par Hamel comme étant $\Delta_i = \frac{(\overline{x}_i - \underline{x}_i)}{\overline{x}_i}$. Pour la méthode multiétape, nous utilisons plutôt les solutions quasi optimales, soit : $\Delta_i = \frac{\left(\overline{\overline{x}}_i - \underline{\underline{x}}_i\right)}{\overline{\overline{x}}_i}$

Le modèle linéaire pour lequel les espaces de solutions explorables ont été générés est celui représenté à la Figure 1 et formalisé dans le modèle 5 de l'annexe 1. Les

billots de bois sont récoltés dans une forêt et peuvent être transformés en papier dans un moulin à papier ou en granules pour fournir de la vapeur. Les deux produits ont le même coût de production et génèrent le même revenu par bille transformée. La demande de papier est inférieure à la capacité de production. L'exercice de planification est réalisé pour trois périodes, pour un total de 37 variables. On cherche à maximiser le profit.

Puisque la demande combinée de granules et de papier dépasse la capacité de production, toute solution optimale devrait impliquer que le maximum de billes doit être récolté en forêt. Toute solution optimale implique également l'atteinte d'un seuil minimum de production de granules afin de maximiser le volume de vente et, de fait, le profit généré. En incluant les quantités de matières transportées, on a cinq (5) variables qui ont une zone assignable limitée à une seule valeur qui ne pourra pas être modifiée par le décideur humain. En ajoutant une tolérance à l'optimalité de 5 %, il devrait devenir possible de réduire le taux d'exploitation de la forêt et la production de granules.

Appliquer l'heuristique triangulaire pour un espace quasi optimal devrait résulter en des zones assignables de variables plus grande pour un nombre significatif de variable modifiables, mais la majorité des solutions retournées par le système à la suite d'une modification par l'utilisateur devraient être sous-optimales. L'utilisation de l'heuristique multiétape devrait augmenter drastiquement le taux de solutions optimales retournées.

3.5. Résultats obtenus

Les détails des mesures sur les espaces de solutions explorables générés pour le modèle linéaire sont présentés au tableau 3.

Tableau 3 – Domaines, zone d'optimalité et zones assignables des variables du modèle 5

Variable	Domaine	Zone d'optimalité	Zone assignable	Gain
Profit	$[0, \infty+]$	$5.04M\$$	$[4.788\ ,5.04\ ]M\$$	5%
Coupe$_i$	$[0{,}168]h$	$168h$	$[142.8, 168]h$	15%
Coupe	$[0{,}504]h$	$504h$	$[478.8, 504]h$	5%
ProductionGranules$_i$	$[0{,}168]h$	$[42{,}168]h$	$[16.8{,}168]h$	15%
ProductionGranules$_T$	$[0{,}504]h$	$[126{,}504]h$	$[100.8{,}504]h$	5%
CamionForetUsine$_i$	$[0{,}1000]K\ m^3$	$[21{,}84]K\ m^3$	$[8.4{,}84]K\ m^3$	15%
CamionForetUsine$_T$	$[0{,}3000]K\ m^3$	$[63{,}252]K\ m^3$	$[50.4{,}252]K\ m^3$	5%
TrainUsineCentrale$_i$	$[0{,}200]K\ Tma$	$[14{,}56]K\ Tma$	$[5.6{,}56]K\ Tma$	15%
TrainUsineCentrale$_T$	$[0{,}600]K\ Tma$	$[42{,}168]K\ Tma$	$[33.6{,}200]K\ Tma$	5%
VenteEnergie$_i$	$[0{,}56]GW$	$[14{,}56]GW$	$[5.6{,}56]GW$	15%
VenteEnergie$_T$	$[0{,}168]GW$	$[42{,}168]GW$	$[33.6{,}168]GW$	5%
CamionForetMoulin$_i$	$[0{,}1000]K\ m^3$	$[0{,}63]K\ m^3$	$[0{,}63]K\ m^3$	0%
CamionForetMoulin$_T$	$[0{,}3000]K\ m^3$	$[0{,}189]K\ m^3$	$[0{,}189]K\ m^3$	0%
ProductionPapier$_i$	$[0{,}168]h$	$[0{,}126]h$	$[0{,}126]h$	0%
ProductionPapier$_T$	$[0{,}504]h$	$[0{,}378]h$	$[0{,}378]h$	0%
CamionMoulinPapeterie$_i$	$[0{,}200]K\ Tma$	$[0{,}10.5]K\ Tma$	$[0{,}10.5]K\ Tma$	0%
CamionMoulinPapeterie$_T$	$[0{,}600]K\ Tma$	$[0{,}31.5]K\ Tma$	$[0{,}31.5]K\ Tma$	0%
VentePapier$_i$	$[0{,}10.5]K\ Tma$	$[0{,}10.5]K\ Tma$	$[0{,}10.5]K\ Tma$	0%
VentePapier$_T$	$[0{,}31.5]K\ Tma$	$[0{,}31.5]K\ Tma$	$[0{,}31.5]K\ Tma$	0%

Tel que prévu, l'introduction d'une tolérance a agrandi les zones assignables pour le profit et la récolte de billots à plus qu'une seule valeur. Pour le profit et l'exploitation totale, on gagne une grandeur de zone d'optimalité équivalant à 5 % de la valeur optimale, et pour l'exploitation à chaque période on obtient une grandeur de zone d'optimalité de 15 % de la valeur optimale.

Pour la production de granules, on obtient une possibilité de réduction de la production de 5 % pour le total des périodes ou de 15 % pour chacune des périodes. Il en est de même pour les variables de transport de billots de bois vers l'usine de granules, le transport des granules au client et la vente de celles-ci au client.

La zone d'optimalité de la valeur des quantités de papier vendues au client par période et au total étant déjà équivalente au domaine de cette variable dans le

modèle, les variables associées n'ont pas vu la taille de leur portés modifiée. Les variables directement contraintes par les quantités vendues, à savoir les quantités de billes de bois vers la papeterie, de papier produit et transporté, de papier transporté vers le client n'ont pas vu la grandeur de leur zone d'optimalité augmenter.

Toutes les autres variables ont vu leur valeur pour le total gagner 5 % de leur valeur maximale en grandeur de zone d'optimalité et les zones d'optimalité des variables par période ont gagné 15 % de la valeur maximale en grandeur.

En ce qui concerne l'optimalité des solutions obtenues, le simple ajout d'une tolérance à la méthode de Hamel a fait apparaître les problèmes prévus. Sur les 72 solutions « extrêmes » initiales produites pour l'interpolation, 52 sont sous-optimales. L'utilisation de l'heuristique multiétape génère 144 solutions de base parmi lesquelles seules 20 étaient sous-optimales. Considérant qu'il y a 20 variables modifiables qui ont vu leur zone assignable s'agrandir, et ce pour un seul extremum, il doit y avoir au moins 20 solutions sous-optimales générées. L'heuristique multiétape a donc généré le nombre minimal de solutions sous-optimales possible.

3.6 Conclusion

Le calcul de l'optimalité fait par l'ordinateur ne se base que sur des valeurs quantifiables alors qu'un utilisateur humain prend aussi en compte des facteurs qualitatifs. L'ajout d'une tolérance à la sous-optimalité permet d'utiliser l'intuition et l'expérience d'un décideur humain pour intégrer la valeur qualitative d'une solution.

L'introduction naïve d'une tolérance à la quasi-optimalité dans la méthode de Hamel introduit un biais significatif vers la sous-optimalité dans les solutions générées. Nous avons donc proposé que la navigation dans l'espace de quasi-optimalité se fasse par une approche multiétape pour gérer le passage de l'optimalité à la quasi-optimalité.

L’application de la méthode de Hamel et *al*, de la méthode de Hamel *et al.* avec tolérance naïve et de l’heuristique multi-étapes sur un modèle simple mais représentatif d’un réseau de création de valeur a permis de confirmer l’introduction du biais vers la sous-optimalité et la capacité de l’approche multiétape d’éliminer celle-ci.

L’approche multiétape permet au système d’inclure des solutions quasi optimales dans la recherche de solution pour contrer les imprécisions du modèle. Elle ne permet cependant pas d’utiliser tout l’espace d’optimalité pour générer des solutions.

4. Proposition d'un algorithme d'exploration par pivot interactif de l'espace des solutions

Tel que vu au chapitre 2, l'espace de solutions généré par la méthode originalement proposée par Hamel *et al.* ne couvre pas la totalité de l'ensemble des solutions optimales existantes pour un problème linéaire. Cela implique qu'il existe des solutions sous-optimales mais potentiellement intéressantes pour l'agent humain qui ne sont pas accessibles par le système. Bien que les améliorations proposées au chapitre 3 permettent à l'utilisateur d'explorer plus de solutions, une partie de l'espace des solutions optimales reste tout de même hors d'atteinte du système.

Par ailleurs, l'algorithme du simplexe offre par sa nature une structure et un moyen de circuler d'un point extrême de l'espace solutions à l'autre. On « pivote » successivement sur des points extrêmes de plus en plus optimaux.

Le présent chapitre propose une modification à l'algorithme du simplexe qui permet de parcourir les points extrêmes de l'espace de solutions en suivant les désirs de l'utilisateur plutôt que la simple poursuite de l'optimalité, le tout permettant à l'utilisateur humain de parcourir la totalité de l'espace des solutions optimales. Ce nouvel algorithme proposé est appelé ***pivot interactif***.

L'idée générale est d'abord proposée, puis le raisonnement menant à l'élaboration d'une nouvelle heuristique de sélection du pivot est élaboré. Un protocole expérimental pour comparer la performance de l'algorithme avec la méthode de Hamel ainsi que les résultats obtenus sont aussi présentés.

4.1. Algorithme général

L'approche proposée démarre avec une première exécution de la version originale du simplexe, afin d'obtenir une solution optimale et le tableau qui l'accompagne. Lors d'une modification par l'utilisateur de la valeur assignée à l'une des variables, le système effectue si possible un pivot pour obtenir une solution dans l'espace d'optimalité qui permettra de générer une nouvelle solution optimale par

combinaison convexe avec la solution courante affichée dans laquelle la valeur assignée à la variable ciblée est la valeur entrée par l'utilisateur.

Le système doit sélectionner le bon pivot pour effectuer une combinaison pertinente. Pour trouver les bons critères de sélection pour $Pivot'(J, \delta)$, détaillons l'opération de pivot telle qu'effectuée dans la méthode du simplexe originale:

Soit une matrice $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ sur laquelle une opération de pivot est effectuée sur a :

$$Pivot\left(a, \begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{bmatrix} 1 & \frac{b}{a} \\ 0 & d - \frac{cb}{a} \end{bmatrix} \quad (5)$$

Appliquons ces résultats à un tableau simplexe $S = \begin{bmatrix} c_j & c_J & v \\ A_{Ij} & A_{IJ} & b_I \\ A_{ij} & A_{iJ} & b_i \end{bmatrix}$ sur laquelle un pivot en A_{IJ} est effectué :

$$Pivot(A_{IJ}, S) = \begin{bmatrix} c_j - \frac{c_J A_{Ij}}{A_{IJ}} & 0 & v - \frac{c_J b_I}{A_{IJ}} \\ \frac{A_{Ij}}{A_{IJ}} & 1 & \frac{b_I}{A_{IJ}} \\ A_{ij} - \frac{A_{iJ} A_{Ij}}{A_{IJ}} & 0 & b_i - \frac{A_{iJ} b_I}{A_{IJ}} \end{bmatrix} \quad (6)$$

Filtrage des pivots invalides

L'opération de pivot étant coûteuse pour de grands tableaux, il est pertinent de faire un filtrage des coordonnées du tableau pour lesquelles un pivot ne placerait pas le tableau sur une solution intéressante. À partir de définitions formalisées aux équations (5) et (6), nous pouvons déterminer le filtrage pour les coordonnées candidates au pivot pour s'assurer que le résultat en soit valide avant d'effectuer l'opération.

1) Afin de s'assurer que le résultat du pivot correspond à une solution réalisable de base, il faut valider qu'aucune contrainte du modèle linéaire original n'est violée, c'est-à-dire que $b' \geq 0$. Formellement :

$$Valide(I,J) \Leftrightarrow b' \geq 0 \quad (7)$$

En reprenant la définition à l'équation *(6)*, pour que le pivot soit valide il faut s'assurer que $\frac{b_i}{A_{IJ}} \geq 0$ pour la rangée sur laquelle le pivot sera effectué et $b_i - \frac{A_{iJ}b_I}{A_{IJ}} \geq 0$ pour les autres. Formellement:

$$Valide(I,J) \Leftrightarrow \forall b_i = \begin{cases} \frac{b_i}{A_{IJ}} \geq 0, & i = I \\ b_i - \frac{A_{iJ}b_I}{A_{IJ}} \geq 0, & i \neq I \end{cases} \quad (8)$$

2) Afin de s'assurer que le résultat du pivot correspond à une solution réalisable de base, il faut aussi s'assurer que toutes les variables de la solution ont une valeur positive, c'est-à-dire que $c' \geq 0$. Formellement :

$$Valide(I,J) \Leftrightarrow c' \geq 0 \quad (9)$$

En reprenant la définition à l'équation *(6)*, si la colonne sur laquelle le pivot sera effectué est dans la base, le pivot est valide puisque $c'_J = 0$, sinon il faut que $c_j - \frac{c_J A_{Ij}}{A_{IJ}} \geq 0$ pour que le pivot soit valide. Formellement :

$$Valide(I,J) \Leftrightarrow \forall b_i = \begin{cases} vrai, & j = J \\ c_j - \frac{c_J A_{Ij}}{A_{IJ}} \geq 0, & j \neq J \end{cases} \quad (10)$$

3) Nous désirons obtenir une solution dont la valeur pour la fonction-objectif soit elle aussi optimale. Puisque l'opération se fait à partir d'un tableau simplexe pour une solution optimale, il faut s'assurer que $v' = v$. Formellement :

$$Valide(I,J) \Leftrightarrow v' = v \quad (11)$$

En reprenant la définition à l'équation *(6)*, pour que le pivot soit valide il faut que $v = v - \frac{c_J b_I}{A_{IJ}}$, donc que $c_J = 0 \vee b_I = 0$. Formellement :

$$Valide(I,J) \Leftrightarrow v - \frac{c_J b_I}{A_{IJ}} = v$$

$$\Rightarrow \qquad \langle arithmétique \rangle$$

$$Valide(I,J) \Leftrightarrow \frac{c_J b_I}{A_{IJ}} = 0$$

$$\Rightarrow \qquad \langle arithmétique \rangle$$

$$Valide(I,J) \Leftrightarrow c_J = 0 \vee b_I = 0 \qquad (12)$$

4) Finalement, il faut s'assurer que l'opération altère le tableau du simplexe. Si la colonne choisie J est une colonne de la base, en reprenant la définition à l'équation (5), $a = 1 \wedge c = 0$ nous obtenons :

$$Pivot\left(a, \begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{bmatrix} 1 & \frac{b}{1} \\ 0 & d - \frac{(0)b}{(1)} \end{bmatrix}$$

$$\Rightarrow \qquad \langle arithmétique \rangle$$

$$Pivot\left(a, \begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (13)$$

Puisqu'un tel pivot ne modifie pas le tableau, il est inutile de l'effectuer.

Sélection du pivot

Puisque nous cherchons une solution pour faire une combinaison convexe en fonction de la valeur attribuée par l'utilisateur à x_J, nous sommes intéressés par la valeur de x'_J. La valeur de la variable x_J est de 0 si la colonne J n'est pas dans la base, sinon à la valeur b_i pour i tel que $A_{iJ} = 1$. Puisque toutes les autres valeurs dans une colonne de la base sont 0, on peut dire formellement :

$$x_J = A_J^T \cdot b \qquad (14)$$

On peut donc prévoir la valeur de la variable x_j pour un pivot en i, j :

$$A'_{iJ} = \begin{cases} 1, & i = I \\ 0, & i \neq I \end{cases} \wedge b'_i = \begin{cases} \dfrac{b_i}{A_{IJ}}, & i = I \\ b_i - \dfrac{A_{iJ} b_I}{A_{IJ}}, & i \neq I \end{cases} \Rightarrow x'_J = \frac{b_I}{A_{IJ}} \quad (15)$$

Ayant posé les principes précédents, nous pouvons établir une méthode de sélection de pivot :

En plus de respecter l'invariant du simplexe, le pivot choisi doit respecter certaines conditions supplémentaires en fonction de l'appartenance de x_J à la base et selon la variation δ de la valeur de x_J.

$J \notin base \Rightarrow x_j = 0$, on peut donc uniquement augmenter la valeur de x_j, qui doit entrer dans la base. Par conséquent, le pivot doit se faire sur J en choisissant I tel que $b'_I > x_J$ et que b'_I soit maximisé.

$J \in base$:

> Si $\delta < 0$, on doit d'abord tenter de sortir x_J de la base, ce qui amènerait sa valeur à 0, soit la plus petite valeur possible. Pour ce faire, il faut choisir I tel que $A_{IJ} = 1$. Si aucun pivot ne respecte les conditions d'invariable, alors on parcourt le reste du tableau pour trouver le I, J tel que $b'_I < x_J$ qui minimise b'_I.
>
> Si $\delta > 0$, il faut maintenir x_J dans la base pour éviter que sa valeur ne descende à 0. Il faut donc choisir I tel que $A_{IJ} \neq 1$.

Si aucun pivot remplissant ces conditions n'existe, alors la variable est à l'extrémité de sa zone d'optimalité.

Sous forme algorithmique, nous obtenons :

Soit $S = \begin{bmatrix} c & v \\ A & b \end{bmatrix}$, un tableau du simplexe pour n variables et m contraintes.
Soit le vecteur X la solution courante affichée.
Soit le vecteur T, la solution de l'état courant du tableau simplexe S.
Soit le naturel J, l'index de la variable ajustée par l'utilisateur.
Soit le réel z, la valeur assignée à la variable par l'utilisateur

Algorithme $PivotInteractif(X: \mathbb{R}^n, T: \mathbb{R}^n, j: \mathbb{N}, z: \mathbb{R}): \mathbb{R}^n$

$\delta \leftarrow z - x_J$
TANT QUE $\delta \cdot (T_J - z) > 0$ FAIRE
    $F \leftarrow X$
    SI $\neg Pivot'(J, \delta)$ ALORS
        RETOURNER T
$\alpha \leftarrow \frac{z - F_J}{T_J - F_J}$
$X \leftarrow \alpha F + (1 - \alpha)T$
RETOURNER X

Algorithme $Pivot'(S: \mathbb{R}^{m+n+1 \times m+1}, J: \mathbb{N}, \delta: \mathbb{R}): \mathbb{B}$

SI J $\notin base$ ALORS
    SI $\delta > 0$ ALORS
        $I \leftarrow -1$
        $p \leftarrow 0$
        POUR $i \in [0, m]$ FAIRE
            SI $\frac{b_i}{A_{iJ}} > p \wedge ValiderPivot(i, J)$ ALORS
                $I \leftarrow i, p \leftarrow \frac{b_i}{AiJ}$
        SI $I > 0$ ALORS
            $Pivot(I, J)$
    RETOURNER $\delta > 0 \wedge I > 0$
$\iota \leftarrow -1$
    POUR $i \in [0, m]$ FAIRE
        SI $A_{iJ} = 1$ ALORS
            $\iota \leftarrow i$
$I \leftarrow \iota$
$Y \leftarrow -1$
$p \leftarrow x_J$
SI $\delta < 0$ ALORS
    POUR $j \in [0, m + n[\setminus base$ FAIRE
        SI $\frac{b_I}{A_{\iota j}} < p \wedge ValiderPivot(i, J)$ ALORS
            $p \leftarrow \frac{b_\iota}{A_{\iota j}}, Y \leftarrow j$
    SI $Y > 0$ ALORS

$Pivot(I, Y)$
SINON
$Pivot(I, Y)$
POUR $i \in [1, m] \setminus I, j \in [0, m + n[\setminus base$ FAIRE
SI $b_\iota - \frac{A_{\iota j} b_i}{A_{ij}} < p \ \wedge ValiderPivot(i, J)$ ALORS
$p \leftarrow b_\iota - \frac{A_{\iota j} b_i}{A_{ij}}, I \leftarrow i, Y \leftarrow j$
SI $Y > 0$ ALORS
$Pivot(I, Y)$
RETOURNER $Y > 0$
POUR $i \in [1, m] \setminus I, j \in [0, m + n[\setminus base$ FAIRE
SI $b_\iota - \frac{A_{\iota j} b_i}{A_{ij}} > p \wedge ValiderPivot(i, J)$ ALORS
$p \leftarrow b_\iota - \frac{A_{\iota j} b_i}{A_{ij}}, I \leftarrow i, Y \leftarrow j$
SI $Y > 0$ ALORS
$Pivot(I, Y)$
RETOURNER $Y > 0$

Algorithme $ValiderPivot(S : \mathbb{R}^{m+n+1 \times m+1}, I : \mathbb{N}, J : \mathbb{N}) : \mathbb{B}$

SI $J \in base \vee (b_I \neq 0 \wedge c_J \neq 0)$ ALORS
RETOURNER $faux$
SI $b_I A_{IJ} < 0$ ALORS
RETOURNER $faux$
POUR $i \in [1, m[\setminus I$ FAIRE
SI $b_i < \frac{A_{iJ} b_I}{A_{IJ}}$ ALORS
RETOURNER $faux$

4.2. Protocole expérimental

Pour qu'un algorithme soit efficace au sein d'un SIP interactif, il doit y avoir une bonne ***rapidité***, c'est-à-dire qu'il doit générer de nouvelles solutions rapidement afin que le système soit utilisable en pratique. Il doit aussi y avoir une bonne ***stabilité***, c'est-à-dire que les variations des valeurs des variables soient les plus petites possible pour éviter de confondre l'utilisateur humain [1].

Pour valider si l'algorithme du simplexe interactif remplit ces critères, un ensemble de six modèles a été choisi. Le modèle 1 et le modèle 2 sont des modèles très

simples permettant de visualiser les solutions et de valider l'algorithme. Le modèle 3, le modèle 4 et le modèle 5 présentent des problèmes d'optimisation simples, le premier de taille très réduite, les deux autres de taille plus importante et de structure similaire, mais l'un présentant un espace d'optimalité plus volumineux. Le modèle 6 présente l'optimisation d'un réseau de création de valeur simpliste sur trois périodes. Le détail des modèles se trouve dans l'annexe 1.

Pour chacun de ces six modèles, la zone d'optimalité de chaque variable a été déterminée par la méthode de Hamel *et al.* Une séquence de petites, moyennes et grandes modifications (respectivement 7%, 45% et 93% de la grandeur de la zone d'optimalité de la variable) sont effectuées sur les variables de chaque modèle. Pour chaque variation, la rapidité est mesurée par le temps de calcul nécessaire en millisecondes. La stabilité est mesurée par le ratio de la distance euclidienne entre la solution originale et la solution obtenue sur la différence de valeur de la variable affectée.

Le code est réalisé en C# et exécuté sur un PC ayant un processeur 64 bits Intel® Core™ i7-4720HQ @ 2.60GHz utilisé sous Windows 10 64 bits.

Pour la rapidité, puisque chaque modification demande d'effectuer une série de pivots, le temps de calcul devrait être sensiblement plus long que ce qui est observé avec l'heuristique triangulaire. De plus, de façon générale, les diminutions de valeur de variables devraient être plus rapides que les augmentations de valeur, puisqu'il est possible de faire une boucle de filtrage sur un nombre de candidats au pivot plus restreint en premier lors de la sélection du pivot.

En ce qui concerne la stabilité, la distance entre la solution générée et la solution courante devrait être supérieure à celle de l'heuristique triangulaire pour de petites modifications, la combinaison se faisant avec une solution sur l'enveloppe de l'espace solutions. Elle fait donc un détour par l'enveloppe au lieu de passer au travers de l'espace solutions. Pour les grandes modifications cependant, la distance entre la solution courante et nouvelle solution générée sera probablement plus

petite, la solution utilisée pour faire la combinaison étant probablement plus proche de la solution courante.

4.3. Résultats

Les résultats sont présentés dans le tableau 4. Lors de l'exécution du programme testant l'algorithme, plusieurs des solutions obtenues par des séries de pivots contenaient des valeurs non numériques inutilisables. Seuls les résultats valides ont été retenus.

Tableau 4 - Résultats du simplexe interactif en termes de rapidité et de stabilité. Les colonnes indiquent le numéro du modèle, le nombre de variables (n) et le nombre de contraintes (m), la grandeur de la modification relative à la taille de la zone d'optimalité (une valeur négative signifie une diminution), le temps de calcul moyen en ticks et en millisecondes pour générer une nouvelle solution et la moyenne de distance euclidienne entre la solution initiale et la solution générée par rapport à la modification. (Les cellules marqués s.o. sont sans objet, c'est-à-dire qu'aucune données valide n'a été obtenue dans ces cas).

modèle			variation	temps moyen		divergence moyenne
	n	m		(ticks)	(ms)	
1	2	3	0.07	5	0.002	1.414213562
			-0.07	1	0.0004	1.414213562
			0.45	0	0	1.414213562
			-0.45	0	0	1.414213562
			0.93	0	0	1.414213562
			-0.93	0	0	1.414213562
2	3	6	0.07	22	0.0087	16.27161043
			-0.07	5	0.002	18.63391271
			0.45	22	0.0087	1.496910398
			-0.45	6	0.0024	3.588094645
			0.93	*s.o.*	*s.o.*	*s.o.*
			-0.93	5	0.002	2.157592418
3	9	12	0.07	1	0.0004	1.632993162
			-0.07	0	0	10.56243555
			0.45	56	0.0222	3.299782682
			-0.45	105	0.0415	1.0335097
			0.93	1	0.0004	1.632993162
			-0.93	*s.o.*	*s.o.*	*s.o.*
4a	53	55	0.07	2	0.0008	1.414213562
			-0.07	*s.o.*	*s.o.*	*s.o.*
			0.45	2	0.0008	1.414213562
			-0.45	*s.o.*	*s.o.*	*s.o.*
			0.93	2	0.0008	1.414213562
			-0.93	*s.o.*	*s.o.*	*s.o.*
4b	53	55	0.07	2	0.0008	1.414213562
			-0.07	1	0.0004	1.414213562
			0.45	2	0.0008	1.414213562
			-0.45	2	0.0008	1.414213562
			0.93	1	0.0004	1.414213562
			-0.93	1	0.0004	1.414213562
6	42	111	0.07	1	0.0004	11.19345902
			-0.07	*s.o.*	*s.o.*	*s.o.*
			0.45	1565	0.6178	3.458985194
			-0.45	8023	3.1671	6.988243224
			0.93	15352	6.0603	1.511243804
			-0.93	8066	3.1841	1.749264703

Les diminutions de valeur (variation négative) ont été plus rapides pour générer une nouvelle solution 5 fois, l'augmentation 4 fois et le temps a été similaire 4 fois. Contrairement à l'intuition initiale, la diminution n'est donc pas significativement plus rapide.

Pour la rapidité de calcul en général, on observe que le temps de calcul dépasse déjà les 6 ms pour le modèle 6 qui présente une structure proche d'un cas réel mais de petite envergure. Ce temps est acceptable, mais il indique que sur des problèmes de dimensions réelles il serait sans doute trop lent. La comparaison avec les temps de calculs sur les mêmes modèles en utilisant la méthode de Hamel *et al.* et l'heuristique triangulaire est présenté au tableau 5.

Tableau 5 - Temps de calculs (en millisecondes) entre l'heuristique triangulaire de Hamel (Hamel-tri) et le simplexe interactif (SI) pour différentes grandeurs de variation de valeur sur une variable.

modèle			Petite variation (7%)		Moyenne variation (45%)		Grande variation (93%)	
	n	m	Hamel-Tri	SI	Hamel-Tri	SI	Hamel-Tri	SI
1	2	3	0	1.18E-03	0	0.00E+00	0	0
2	3	6	0	3.65E-03	0	3.95E-03	0	1.97E-03
3	9	12	1.32E-04	7.90E-05	1.32E-04	3.04E-02	0	3.95E-04
4a	53	55	1.32E-04	7.90E-04	1.32E-04	7.90E-04	1.32E-04	7.90E-04
4b	53	55	1.32E-04	5.92E-04	1.32E-04	7.90E-04	1.32E-04	3.95E-04
6	42	111	1.32E-04	3.95E-04	0	1.04E+00	1.32E-04	4.62E+00

Finalement, en ce qui a trait à la stabilité, la méthode présente une stabilité similaire à la méthode par heuristique triangulaire pour de grandes modifications mais est très instable pour de petites modifications, tel que prévu. La comparaison avec les résultats sur les mêmes modèles en utilisant la méthode de Hamel et l'heuristique triangulaire est présentée dans le tableau 6.

Tableau 6 - Comparaison de la stabilité entre l'heuristique triangulaire de Hamel (Hamel-tri) et le simplexe interactif (SI) pour différentes grandeurs de variation de valeur sur une variable.

modèle			Petite variation (7%)		Moyenne variation (45%)		Grande variation (93%)	
	n	m	Hamel-Tri	SI	Hamel-Tri	SI	Hamel-Tri	SI
1	2	3	1.41	1.41	1.41	1.41	1.41	1.41
2	3	6	1.76	18.04	1.76	3.07	1.76	2.16
3	9	12	2.15	8.78	2.15	2.33	2.15	1.63
4a	53	55	1.14	1.41	1.14	1.41	1.14	1.41
4b	53	55	3.05	1.41	3.05	1.41	3.05	1.41
6	42	111	1.01	11.19	1.01	4.05	1.01	1.63

4.4. Conclusion

En théorie, l'algorithme du pivot interactif permet à l'utilisateur d'explorer la totalité de l'espace des solutions optimales. En pratique cependant, bien qu'il ait une bonne rapidité pour un nombre relativement petit de variables et de contraintes, celle-ci diminue drastiquement pour les plus gros problèmes. Il est peu probable qu'il puisse demeurer interactif pour des modèles de taille industrielle (de trois à quatre ordres de magnitude plus complexes que les modèles évalués). En termes de stabilité, il est très bon sur certains problèmes ayant une topologie particulière mais est inférieur à l'heuristique triangulaire pour les modèles de grandeurs et de topologies industrielles.

5. Proposition d'algorithmes avec cartographie complète de l'espace d'optimalité pour son exploration interactive

La méthode proposée par Hamel *et al.* consiste à « cartographier » l'espace des solutions optimales par un ensemble de solutions qui associe à chaque variable modifiable par l'utilisateur une solution optimale qui minimise cette valeur et une solution qui maximise cette valeur. Par la suite, lorsque l'utilisateur change la valeur assignée à l'une des variables, la solution associée à l'extremum visé pour cette variable est combinée avec la solution affichée pour en créer une nouvelle en temps réel.

Malheureusement, l'espace contenu par le polytope dont les points extrêmes correspondent à l'ensemble de solutions généré par la méthode de Hamel *et al.* ne couvre qu'une partie de l'espace des solutions optimales. La « cartographie » de l'espace solutions est donc incomplète. L'heuristique triangulaire ne permettant pas de générer des solutions en dehors de l'espace cartographié, certaines solutions optimales ne peuvent pas être explorées par l'utilisateur humain.

Pour avoir accès à un plus grand ensemble de solutions optimales, il faut utiliser plus de points extrêmes, ce qui demande une méthode de sélection des solutions à combiner qui puisse s'accommoder de n'importe quel polytope convexe, tout en conservant les propriétés de rapidité et de stabilité.

Une telle méthode permettrait l'exploration de la totalité de l'espace des solutions optimales, à condition d'en obtenir la totalité des points extrêmes formant son enveloppe convexe. Des travaux précédents traitent de l'énumération des points extrêmes [7] mais ne se concentrent pas sur les points optimaux. D'autres travaux se sont concentrés sur l'espace d'optimalité [23], mais la méthode proposée n'énumère pas les points extrêmes.

Le présent chapitre propose un algorithme de navigation dans un polytope à n-dimension. Pour complémenter cet algorithme, un algorithme d'énumération des points extrêmes de l'espace des solutions optimales est présenté. Les performances de ces algorithmes sont comparées à celles de la méthode de Hamel avec l'heuristique triangulaire

5.1. Exploration d'un espace délimité par un nuage de points extrêmes

Appelons ***point courant*** le point de l'hyperespace correspondant à la solution actuellement choisie par l'utilisateur.

L'approche proposée consiste à générer une nouvelle solution en combinant le point courant avec le point extrême le plus proche qui remplit les conditions suivantes :

- la solution se trouve dans la direction de la modification faite par l'utilisateur (i.e. si l'utilisateur augmente la valeur de x, alors la solution avec laquelle on combine doit avoir une valeur de x plus grande que la solution courante);
- la succession du choix des points extrêmes pour la combinaison lors de la modification d'une variable jusqu'à sa valeur extrême ne doit pas provoquer d'oscillation dans les valeurs des autres variables (i.e. si l'utilisateur augmente la valeur d'une variable x, une variable y ne doit pas tantôt augmenter, tantôt diminuer.

Pour effectuer la recherche de ce point extrême, l'algorithme proposé utilise un filtrage par boîte de bornage alignée sur les axes (en anglais « Axis-aligned bounding box » ou ***AABB***). L'AABB définit un espace en fixant les valeurs minimale et maximale sur chaque dimension (dans notre cas, les variables modifiables) pour former une boîte. L'AABB permet de filtrer rapidement les points qui en sont exclus en comparant directement les valeurs des variables avec les valeurs définies comme limites de l'AABB [24].

Il existe plusieurs méthodes pour définir une AABB [24]. Dans notre cas, nous utilisons les points minimum et maximum de l'AABB. Pour les déterminer, nous utilisons le point courant et un ***point terminal***. Le point terminal est le point extrême optimisant la variable modifiée dans la direction souhaitée le plus près de la solution

courante. Pour définir l'AABB par son minimum et son maximum, nous utilisons les valeurs de chaque variable dans le point courant et le point terminal pour obtenir le minimum et le maximum pour chaque variable.

Afin d'accélérer la recherche du point extrême qui formera l'AABB avec le point courant, les points extrêmes du polytope formant l'espace des solutions optimales sont indexés *a priori* par extremum de la zone d'optimalité de chaque variable modifiable. Cette approche est similaire à celle de Hamel, à la différence que toutes les solutions optimisant la variable sont indexées et non seulement la première trouvée. Le carré de la distance euclidienne permet ensuite de trouver rapidement la solution la plus proche de la solution courante à l'intérieur du sous-ensemble de solution indexées.

En filtrant les points extrêmes en dehors du sous espace défini par l'AABB entre le point courant et le point terminal utilisés pour les combinaisons convexes, nous obtenons une plus grande stabilité sur les valeurs des autres variables durant la manipulation. La figure 25 illustre le principe dans un scénario où l'utilisateur augmente la valeur de x. Dans le premier cas non borné, la variation de y est d'abord décroissante, puis croissante et finalement décroissante à nouveau. Dans le deuxième cas (borné), la variation de y est toujours décroissante.

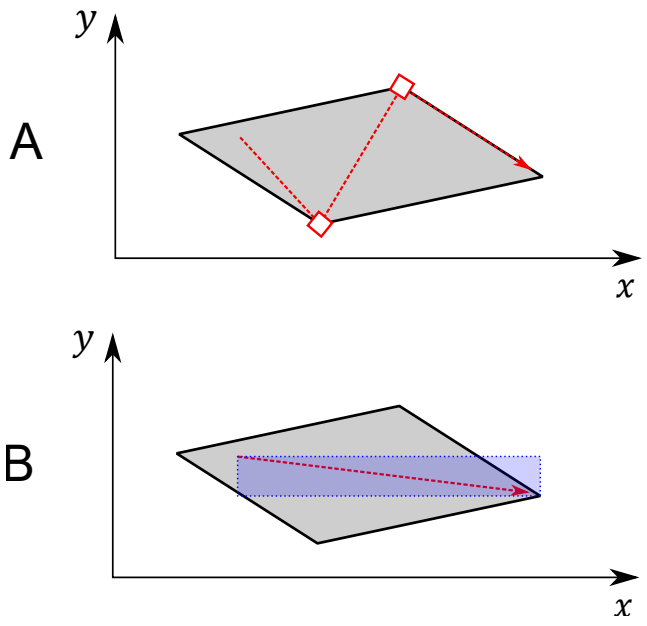

Figure 25 - Comparaison d'une recherche de parcours de points extrême avec et sans bornage. Le cas A n'est pas borné et les variations de la valeur en y sont moins stables pour une variation en x que pour le cas borné B (où l'AABB est indiquée en bleu).

Nous proposons l'algorithme « N-Dimensional Navigation Direction » (direction de navigation à *n*-dimension, ***NDND***), qui procède comme suit :

1) on trouve le point terminal, soit parmi les points extrêmes indexés associés à la direction de la modification de la variable, soit celui qui est le plus proche du point courant;
2) on construit une AABB avec le point courant et le point terminal;
3) on trouve le point extrême du polytope de l'espace des solutions optimales à l'intérieur de l'AABB le plus près du point courant;
4) on combine le point courant avec le point obtenu en 3) jusqu'à ce que $\alpha = 1$;
5) on pose la solution trouvée en 3) comme point courant;
6) on répète 3) à 5) jusqu'à atteindre la limite de l'AABB ou la valeur désirée par l'utilisateur.

Sous forme algorithmique, nous obtenons :

Soit P, l'ensemble des points extrêmes du polytope de l'espace des solutions
Soit i, l'index de la variable à ajuster.
Soit I, un index associant le double de l'index d'une variable à l'ensemble des points extrêmes maximisant la valeur de celle-ci et le double de l'index + 1 à l'ensemble des points extrêmes minimisant la valeur de la variable.
Soit v, la valeur assignée à la variable à ajuster.
Soit c, le point courant
Soit u, le nombre de points extrêmes du polytope
Algorithme $NDND\ (P : \mathbb{R}^{n\times u}, I: \mathbb{N} \mapsto: \mathbb{R}^{n\times\mathbb{N}}, i : \mathbb{N}, v : \mathbb{R},\ c : \mathbb{R}^n)$

$optimaux : \mathbb{R}^{n\times\mathbb{N}}$
$d : \mathbb{R}$
$max : \mathbb{R}$
$t : \mathbb{R}^n$
$candidats: \mathbb{R}^{n\times\mathbb{N}}$

SI $v > c[i]$ ALORS
    $optimaux \leftarrow I[2i]$
SINON
    $optimaux \leftarrow I[2i+1]$
$t \leftarrow argmin\{p \in optimaux\}\ Distance(solutionCourante, p)$
$candidats \leftarrow \{p \in P \mid c[k] \leq\ p[k] \leq t[k] \vee\ t[k] \leq\ p[k] \leq c[k]\ \forall k \in [0..n]\}$
$v \leftarrow argmin\{p \in candidats\} Distance(solutionCourante, p)$
$\alpha \leftarrow \dfrac{v - t[i]}{t[i] - c[i]}$
RETOURNER $\alpha c + (1-\alpha)t$

Protocole expérimental pour valider l'algorithme NDND

Pour tester l'algorithme NDND, un ensemble de six modèles a été choisi. Le modèle 1 et le modèle 2 sont des modèles très simples permettant de visualiser les solutions et de valider l'algorithme. Le modèle 3, le modèle 4 et le modèle 5 présentent des problèmes d'optimisation simples, le premier de taille très réduite, les deux autres de tailles plus importantes et de structures similaires, mais l'un présentant un espace d'optimalité plus volumineux. Le modèle 6 présente l'optimisation d'un réseau de

création de valeur simpliste sur trois périodes. Le détail des modèles se trouve dans l'Annexe 1 : Modèles linéaires utilisés pour évaluer les performances des algorithmes.

Pour chacun des modèles, un ensemble de solutions optimales est généré par la méthode de Hamel. Puis, une séquence de petites, moyennes et grandes modifications (respectivement 7%, 45% et 93% de la zone d'optimalité de la variable) sont effectuées par l'heuristique triangulaire et par NDND. Pour chaque test, la rapidité et la stabilité sont mesurées. La rapidité est mesurée en temps de calcul (millisecondes). La stabilité est mesurée par le ratio de la distance euclidienne entre la solution originale et la solution obtenue sur la différence de valeur de la variable affectée.

Au vu de la plus grande complexité des calculs, l'algorithme NDND devrait être plus lent que l'heuristique triangulaire de Hamel. En revanche, puisque le point terminal de NDND devrait être un point extrême plus proche du point courant (ou au pire le même) que le point extrême utilisé par l'heuristique triangulaire, la stabilité de NDND devrait être meilleure ou égale.

Résultats

Comme prévu, le temps de calcul pour l'algorithme NDND est plus important que celui de l'heuristique triangulaire. Cependant, les temps de calculs de NDND demeurent très acceptables, avec un ordre du centième de milliseconde pour un modèle à 53 variables. Les résultats de l'expérimentation pour la rapidité sont présentés dans le tableau 7.

Tableau 7 - Comparaison des temps de génération de solution pour l'heuristique triangulaire et l'algorithme NDND

modèle			petite modification (7%)		moyenne modification (45%)		grande modification (93%)	
	n	m	Hamel-Tri	Hamel-NDND	Hamel-Tri	Hamel-NDND	Hamel-Tri	Hamel-NDND
1	2	3	0*	3.95E-04	0*	5.26E-04	0*	2.63E-04
2	3	6	0*	3.95E-04	0*	3.95E-04	0*	3.95E-04
3	9	12	1.32E-04	6.58E-04	1.32E-04	3.95E-04	0*	6.58E-04
4a	53	55	1.32E-04	1.21E-02	0.00E+00	1.24E-02	1.32E-04	1.07E-02
4b	53	55	1.32E-04	5.66E-03	1.32E-04	5.66E-03	1.32E-04	5.66E-03
6	42	111	1.32E-04	2.89E-03	1.32E-04	2.76E-03	1.32E-04	2.63E-03

* < 3.95E-07 ms

Contrairement à ce qui était prévu, NDND ne présente pas d'amélioration significative de la stabilité. Seul le modèle 2 présente un gain de 20 %. Les résultats de l'expérimentation pour la stabilité sont présentés dans le tableau 8. Aucune explication n'a été trouvée pour cette différence.

Tableau 8 - Comparaison des stabilités de génération de solution par l'heuristique triangulaire et l'algorithme NDND

modèle			petite modification (7%)		moyenne modification (45%)		grande modification (93%)		Ratio NDND/Tri
	n	m	Tri	NDND	Tri	NDND	Tri	NDND	
1	2	3	1.41	1.41	1.41	1.41	1.41	1.41	1.00
2	3	6	1.76	1.41	1.76	1.41	1.76	1.41	0.80
3	9	12	2.15	2.98	2.15	2.98	2.15	2.98	1.38
4a	53	55	1.01	1.41	1.01	1.41	1.01	1.41	1.40
4b	53	55	1.14	1.41	1.14	1.41	1.14	1.41	1.23
6	42	111	3.05	3.64	3.05	3.64	3.05	3.64	1.19

L'algorithme NDND offre des performances légèrement inférieures à l'heuristique triangulaire dans le cas d'un espace généré par la méthode de Hamel. Elle n'est

donc pas indiquée pour cette situation. Cependant, elle demeure suffisamment rapide et stable pour être utilisée dans le cas d'un espace de solutions défini par un polytope convexe.

5.2. Agrandissement de la portion explorable de l'espace solutions

Maintenant équipés d'une méthode pour explorer un polytope convexe défini par ses points extrêmes, nous pouvons nous intéresser à augmenter le nombre de points extrêmes générés *a priori* par le système.

Soit M, un programme linéaire tel que :

$$\begin{array}{rl} min & c^T x_i \\ M = s.o: & Ax_i \leq b_i \\ & x_i \geq 0 \end{array}$$

Définissons S un tableau SIMPLEXE tel que :

$$S(M) = \begin{array}{ccc} c & \langle 0 \rangle_m & v \\ A & I_m & b \end{array}$$

Soit P le polytope englobant l'espace des solutions optimales de M. Afin de trouver la totalité de l'espace des solutions optimales, il faut en énumérer tous les points extrêmes de P. Yamada *et al.* discutent du problème de l'énumération des points extrêmes d'un polytope défini par un ensemble de contraintes linéaires [7]. Ils notent deux obstacles majeurs dans la résolution de ce problème, soient l'instabilité numérique et la dégénérescence des polytopes définis par un ensemble de contraintes.

Le principal effet de l'instabilité numérique rapporté par Yamada *et al.* est la multiplication des points extrêmes générés lors du parcours du polytope par séquence de pivots. Ils soulignent aussi que si un point extrême P respecte plus de n égalités, l'opération de pivot sur celui-ci peut générer un point extrême « voisin » qui est en fait le même point extrême. Ces deux obstacles les mènent à la conclusion

que l'opération de pivot ne peut servir à l'énumération des points extrêmes d'un polytope définissant l'espace des solutions d'un programme linéaire. Ils proposent l'algorithme Extreme2, une approche par compressibilité de la matrice A.

Puisque l'algorithme Extreme2 se concentre d'abord sur l'espace des solutions, il faut préparer le système de contraintes. Cette préparation peut se faire en obtenant une solution optimale par une méthode classique (simplexe, ellipsoïde, Karmarkar), puis en injectant une contrainte forçant la fonction-objectif à avoir la valeur optimale. Il existe aussi un algorithme proposé par Kantor pour réduire une matrice de contraintes à la matrice de l'espace d'optimalité [23].

Proposition d'un algorithme pour la couverture complète de l'ensemble des solutions optimales d'un modèle linéaire

Le cadre pratique de l'application visée nous permet cependant de contourner les limites à la succession de pivots imposées par l'instabilité numérique. Il est possible de fixer un seuil de précision en deçà duquel deux solutions sont considérées identiques. Par exemple, un seuil de précision de l'ordre de $1x10^{-6}$ correspond à une précision de l'ordre du cent sur un budget se calculant en dizaine de milliers de dollars ou de l'ordre du gramme pour un transport se mesurant en tonnes. L'application d'un tel seuil permet de filtrer les points extrêmes redondants lors d'une séquence de pivots pour continuer l'exploration des limites de l'espace optimal à partir d'une solution trouvée par le simplexe.

Nous proposons ***LRRR*** (de l'anglais « Linear Redundancyless Recursive Research »), un algorithme de recherche en profondeur d'abord par récursivité pour un problème linéaire filtrant les redondances. L'idée fondamentale de LRRR est de tenter un pivot sur chaque cellule de la matrice A pour chercher de nouvelles solutions optimales. Lorsque la solution générée par le pivot est valide, optimale et nouvelle, on démarre une nouvelle recherche sur la matrice obtenue.

Une telle recherche pouvant être assez longue sur un cas réel, il est nécessaire de fixer certaines heuristiques pour éviter d'effectuer des pivots vers des solutions invalides ou sous-optimales. Trois heuristiques sont présentées pour LRRR, soit éviter les pivots sur une valeur nulle (no null pivot), la conservation de la valeur de b_i (b-preserve) et la conservation de la valeur totale de la solution (v-preserve).

No null pivot consiste à pivoter sur A_{ij} seulement si $A_{ij} \neq 0$. En effet, lors de la normalisation de la ligne i durant l'opération de pivot, il faut diviser la rangée par A_{ij}, or si $A_{ij} = 0$ cela implique de faire des divisions par 0, une violation des règles fondamentales de l'arithmétique.

b-preserve consiste à pivoter sur A_{ij} seulement si $b\_i/A\_ij > 0$. Cette heuristique est encore une fois liée à la normalisation de la rangée i. Si $b\_i/A\_ij < 0$, on sait que la solution générée par le pivot ne respecte pas l'invariant du simplexe et que la solution ne respecte donc pas la contrainte exprimée par la rangée i.

v-preserve consiste à ne pivoter sur A_{ij} que si $b_i = 0 \vee c_j = 0$. Puisque l'opération de pivot applique l'opération $\forall S_{kl}: k \neq i | S'_{kl} = S_{kl} - S_{kj}S_{il}$, ce qui implique que $v' = v - b_i c_j$. Or, afin de ne retourner que les points optimaux, v ne doit pas être modifié. Pour que $v' = v$, il faut que $(b_i c_j = 0) \equiv (b_i = 0 \vee c_j = 0)$.

Considérant la taille du tableau simplexe passé en paramètre et du polytope généré, ces éléments sont passés par référence au niveau de récursion suivant. Par conséquent, l'algorithme tient une pile de coordonnées de pivot pour pouvoir rétablir le tableau simplexe dans son état précédent après avoir traité une possibilité de pivot.

Sous forme algorithmique, nous obtenons :

Soit S, un tableau simplexe de forme $\begin{bmatrix} c & \langle 0 \rangle_m & v \\ A & I_m & b \end{bmatrix}$
Algorithme $LRRR(S : \mathbb{R}^{n+1 \times n+m+1})$

$poly : polytope \leftarrow \emptyset$
$p : pile\ de\ (paires\{\mathbb{N}, \mathbb{N}\}) \leftarrow \emptyset$
$SIMPLEX(S)$
$poly.ajouter(S.solution)$
$Sonder(S, poly, p)$
RETOUTNER $poly$

Algorithme
$Sonder(S : \mathbb{R}^{n+1 \times n+m+1}, poly : polytope, p : pile\ de\ (paires\{\mathbb{N}, \mathbb{N}\}))$

POUR TOUT $S_{ij} \in (A)$ FAIRE
    SI $S_{ij} \neq 0 \wedge b_i / S_{ij} < 0 \wedge (b_i = 0 \vee c_j = 0)$ ALORS
        $p.Empiler(Pivot(i, j))$
        SI $b \geq 0 \wedge \neg p.contient(S.solution)$ ALORS
            $poly.ajouter(S.solution)$
            $Sonder(S, poly, p)$
        $Pivot(p.depiler)$

Protocole expérimental pour valider la cartographie de l'espace de solutions

Les 6 modèles utilisés pour comparer NDND et l'heuristique triangulaire sont repris pour valider l'algorithme LRRR. Leur structure simple permet de déterminer trivialement la topologie de leur espace de solutions et de leur espace de solutions optimales afin de servir de référence pour la validation des résultats de LRRR. Les calculs ont été effectués sur un ordinateur utilisant un processeur Intel® CORE™ i7-2630QM @2.00GHz dans un environnement Windows 8.1.

Il est à noter que le cadre expérimental est limité par la capacité en mémoire de l'ordinateur. En effet, si on reprend le réseau de production de valeur du modèle 6, un réseau très simple, et qu'on planifie sur 52 périodes au lieu de 3, l'énumération

des points extrêmes produit en tout 2^{104} points extrêmes, chacun contenant $728 > 2^9$) variables à virgule flottante (2^4 octets pour un nombre flottant à double précision). Un tel problème trivial occupe donc plus de 2^{117} octets, ce qui dépasse largement la capacité mémoire d'un ordinateur moderne.

Afin de limiter la mémoire utilisée par la recherche récursive lors de l'implémentation, chaque opération de pivot effectuée durant la recherche pousse sur une pile le pivot inverse. Ainsi, un retour arrière dans l'arbre de recherche se fait en dépilant une opération de pivot.

Le nombre de point extrêmes à sonder étant élevé, il faut s'attendre à ce que les temps de calculs soient relativement longs. LRRR devrait cependant retourner la totalité des points extrêmes de l'espace des solutions optimales.

Résultats obtenus pour la cartographie de l'espace de solutions

L'algorithme est expérimentalement correct puisqu'il trouve tous les points extrêmes optimaux. De plus, il le fait dans un temps acceptable sur une machine standard. Le nombre de retours arrière en fonction du nombre de PE est variable. Il est élevé sur les petits problèmes, impliquant que certains points extrêmes sont visités plusieurs fois. Il est cependant peu élevé sur les plus gros problèmes. Les heuristiques de filtrage semblent donc efficaces pour les cas avec de nombreux points. Ces résultats sont présentés dans le tableau 9.

Tableau 9 - Résultats expérimentaux de LRRR. Pour chaque modèle, le nombre de variables (n), de contraintes(m) de points extrêmes réalisables (PE) et optimaux (PEO) est indiqué. Les résultats du test pour chaque modèle contiennent le nombre de point extrêmes optimaux trouvés, le temp de calcul en secondes et le nombre de points extrêmes sondés.

Modèles					Résultats		
	n	m	Nb. de PE	Nb. de PEO	PEO trouvés	Temps (s)	PE sondés
1	2	3	5	2	2	0,02	4
2	3	6	14	8	8	0,01	48
3	9	12	64	8	8	0,01	60
4a	53	55	4,5036E+15	52	52	0,73	2806
4b	53	55	4,5036E+15	1326	1326	20,48	72826
6	42	111	1000	64	64	8,25	4704

Comparaison des algorithmes LRRR et Hamel

Le but de l'élaboration de l'algorithme LRRR est d'augmenter la couverture de l'espace des solutions optimales. Pour mesurer l'augmentation de cette couverture, on ne peut se fier à la comparaison des hypervolumes des espaces de solutions optimales obtenus puisque pour cela, il faut des espaces de même dimensionalité. Tout d'abord, l'espace des solutions optimales est toujours de dimensionnalité inférieure à l'espace réalisable, son volume est toujours nul (cela reviendrait à mesurer le volume de la face d'un cube) S'Il est possible d'en mesurer la grandeur pour une dimensionnalité moindre (pour reprendre l'exemple de la face d'un cube, on peut en mesurer la surface), il n' y a pas de garantie que l'espace de solutions optimales retourné par la méthode de Hamel soit de la même dimensionalité que l'espace de toutes les solutions optimales. Par exemple, si les coins de l'AAB englobant l'espace des solutions sont des points extrêmes de celui-ci, la méthode de Hamel retourne un segment de droite en une seule dimension, et ce, peu importe la dimensionnalité de l'espace des solutions optimales. On ne peut donc pas établir avec certitude une dimensionnalité commune connue pour faire une comparaison valide entre les espaces de solutions optimales obtenus par les deux méthodes.

Cependant, la comparaison peut être effectuée en fonction du nombre de points extrêmes optimaux distincts trouvés (certains points extrêmes peuvent optimiser plusieurs variables et il y a donc des redondances dans la matrice d'optimalité de Hamel). Cette comparaison est présentée au tableau 10.

Tableau 10 - Comparaison du nombre de points extrêmes trouvés par les algorithmes Hamel et LRRR

Modèles					PEO trouvés	
	n	m	Nb. de PE	PEO	LRRR	Hamel
1	2	3	5	2	2	2
2	3	6	14	8	8	3
3	9	12	64	8	8	5
4a	53	55	4.50E+19	52	52	105
4b	53	55	4.50E+19	1326	1326	105
6	42	111	1000	64	64	13

Le nombre de points extrêmes distincts trouvés par Hamel est généralement plus bas que pour LRRR. Il y a cependant une exception pour le modèle 4a, où la méthode de Hamel trouve 105 points extrêmes. L'exception peut s'expliquer par l'ajout de points extrêmes par dégénérescence du polytope lors de l'ajout de la contrainte forçant l'optimalité.

5.3. Conclusion

L'algorithme NDND offre une performance acceptable pour explorer un polytope convexe défini par ses points extrêmes, mais il demeure cependant moins efficace que l'heuristique triangulaire si l'espace solutions est cartographié selon la méthode de Hamel. Il n'est donc intéressant que si l'espace solutions est cartographié autrement pour contenir plus de points extrêmes.

L'algorithme LRRR permet de cartographier la totalité de l'espace solutions d'un programme linéaire. Cet espace est cependant trop gourmand en mémoire pour une application réelle et se limite à des modèles jouets. Il n'est donc pas un bon candidat pour remplacer la méthode de Hamel dans un cadre d'utilisation industriel.

Par conséquent, la méthode de Hamel et l'heuristique triangulaire demeurent l'approche à privilégier pour les cas réels. Le grand nombre de points extrêmes délimitant de tels espaces ne laisse pas entrevoir dans un avenir proche un meilleur compromis entre la couverture de l'espace solutions, la rapidité de calcul et la taille mémoire de la solution. Il semble donc que, pour l'instant, la recherche doive se tourner vers des approches ne faisant pas appel à l'énumération de points extrêmes.

Conclusion

Le problème de la planification optimale des tâches au sein d'un réseau de production de valeur demande la prise en charge d'un grand nombre de données et de calculs qui dépasse la capacité humaine. Ce problème de planification peut être exprimé sous la forme d'un problème d'optimisation linéaire. Il existe des algorithmes, notamment celui du simplexe, qui permettent aux ordinateurs de résoudre ce type de problèmes [5]. L'ordinateur n'est cependant pas sans limites technologiques, les solutions obtenues par l'ordinateur ne prennent en compte que les facteurs quantitatifs et l'humain est souvent méfiant envers une solution sortie d'une machine s'il ne comprend pas totalement la solution ou la machine [15].

Un outil comme Logilab, qui présente les données de la solution de façon claire, facilite dans une certaine mesure la compréhension de la solution [12]. Cette meilleure compréhension de la solution ne garantit cependant pas son acceptation. Les systèmes à initiative partagée, qui font travailler en collaboration l'humain et la machine pour résoudre un problème, permettent de jumeler les forces de l'ordinateur et de l'humain pour obtenir de meilleures solutions et augmenter le niveau de confiance de l'humain envers la solution obtenue [2]. L'utilisation des propriétés géométriques des systèmes d'inéquations a permis la mise au point d'un tel système pour la planification optimale des tâches au sein d'un réseau de production de valeur. Ce système crée un espace de solutions optimales que l'humain peut explorer pour obtenir une solution optimale qui lui convienne [1].

Le système de Hamel *et al.* a cependant des limites. Premièrement, il ne présente pas les données de la solution en contexte comme Logilab, ce qui impose un charge mentale supplémentaire à l'humain pour comprendre ce qu'il fait. De plus, l'espace de solutions explorable est limité aux solutions dont l'optimalité est mesurée uniquement par les critères quantitatifs gérés par l'ordinateur. Finalement, l'espace de solutions explorables ne couvre pas toutes les solutions optimales existantes pour le problème. Nous avons donc proposé une série d'améliorations aux systèmes à initiative partagée humain-ordinateur pour l'optimisation des systèmes linéaires.

Tout d'abord, nous avons proposé une nouvelle interface humain-machine pour améliorer l'expérience et la performance de l'utilisateur lors de la recherche de solution. Nous avons déterminé un utilisateur cible, le partenaire industriel gestionnaire d'entreprise. Une analyse rigoureuse des caractéristiques, des besoins et des attentes de l'utilisateur a confirmé que le logiciel Logilab offrait une présentation des données adaptée à l'utilisateur. À partir de ce modèle, des itérations ont été réalisées pour intégrer la présentation des informations de zone d'optimalité pour les variables ainsi que les mécanismes d'interactivité pour l'exploration de l'espace de solutions. Une version d'essai interactive de cette interface a été implémentée pour en valider le fonctionnement. Cette interface s'est révélée utile, mais ne réglait qu'un des problèmes du système proposé par Hamel *et al.*

Nous avons ensuite proposé une approche permettant une tolérance quant à l'optimalité des solutions, c'est-à-dire permettre à l'utilisateur d'explorer des solutions légèrement sous-optimales du point de vue du système automatisé. Nous avons tout d'abord envisagé une simple expansion de l'espace de solutions explorable. Cette approche naïve pose cependant un problème lors de la génération de nouvelles solutions, puisqu'elle ne retourne pas nécessairement une solution optimale même s'il serait possible de le faire. Une double cartographie du sous-espace des solutions explorables, soit un espace optimal et un espace quasi optimal, permet l'introduction d'une heuristique multiétape garantissant une solution optimale lorsque possible. L'implémentation de cette approche en a confirmé le bon fonctionnement, autant en termes de rapidité que de stabilité. Cette approche permet de valoriser le jugement qualitatif de l'humain dans le processus de décision et d'agrandir l'espace des solutions explorables. Elle ne permet cependant pas de garantir la couverture complète de l'espace des solutions optimales par l'espace des solutions explorables.

Pour agrandir l'espace des solutions explorables et tenter la couverture complète de l'espace des solutions optimales, deux approches ont été envisagées, soit l'abandon

de l'étape de cartographie de l'espace des solutions optimales et sa cartographie complète.

Pour naviguer dans l'espace d'optimalité sans en cartographier les points extrêmes, Il est possible de modifier l'algorithme du simplexe pour le rendre interactif. Nous avons proposé l'algorithme du pivot interactif, qui consiste à maintenir le tableau simplexe dans son état optimal et d'effectuer des pivots au-devant des modifications que l'utilisateur effectue pour générer une nouvelle solution par combinaison convexe. Si nous avons démontré que cette approche est théoriquement correcte pour permettre l'exploration de la totalité de l'espace des solutions optimales, elle ne présente pas de garantie de performance en termes de rapidité de calcul pour que son utilisation dans un système interactif en temps réel soit envisageable.

Nous avons donc proposé une autre approche, soit la cartographie de la totalité de l'espace des solutions optimales. Cette approche demande à la fois d'établir une nouvelle méthode de navigation dans les points extrêmes trouvés et un algorithme de recherche de tous les points extrêmes optimaux.

Nous avons proposé l'algorithme NDND (*N-Dimensional Navigation Direction*) pour l'exploration d'un espace de solutions défini par les points extrêmes d'un polytope convexe quelconque. Cet algorithme prend successivement le point extrême le plus près du point en filtrant les points extrêmes qui engendreraient de l'instabilité dans le parcours. Cet algorithme présente des performances de stabilité et de temps de calcul expérimentales comparables à l'heuristique triangulaire.

L'algorithme LRRR (*Linear Redundancyless Recursive Research*) permet d'énumérer la totalité des points extrêmes du polytope définissant l'espace des solutions optimales par une recherche en profondeur d'abord à partir d'un premier point extrême obtenu par l'algorithme du simplexe. Trois heuristiques s'assurent de n'obtenir que des points extrêmes correspondant à des solutions optimales réalisables tandis qu'une validation tenant en compte l'instabilité numérique prévient la redondance de points. Cet algorithme est juste, il est cependant irréaliste de

l'appliquer à des problèmes réels considérant que le nombre de points extrêmes optimaux pour un problème réel est au-delà de la capacité mémoire des ordinateurs actuels.

L'interface proposée au chapitre 2 permet une meilleure compréhension des données manipulées par l'utilisateur humain, ce qui lui permet une meilleure collaboration avec le système informatique pour l'élaboration interactive d'un plan tactique d'opérations. L'ajout d'une tolérance à l'optimalité permet aussi à l'expérience et l'intuition de l'utilisateur d'avoir plus de poids dans le plan obtenu. Il n'est cependant pas envisageable de couvrir la totalité de l'espace des solutions optimales avec les ordinateurs actuels.

Autres pistes explorées

Dans la poursuite de l'agrandissement de l'espace de solutions explorables pour couvrir la totalité de l'espace des solutions optimales, d'autres approches pourraient être explorées. D'une part, des algorithmes d'optimisation linéaires théoriquement plus performants que le simplexe pourraient être étudiés plus avant, autant pour accélérer la cartographie de l'espace solutions que dans l'optique de les rendre interactifs pour la navigation. Une nouvelle approche inspirée de la simulation physique pourrait aussi être envisagée.

Méthode des ellipsoïdes

La méthode des ellipsoïdes fût introduite par L.G. Kachiyan en 1979 comme première méthode de résolution d'un modèle d'inégalité linéaire en temps strictement polynomial [25]. Contrairement au simplexe, qui repose sur l'aspect combinatoire des points extrêmes du polytope formant l'espace solutions, Cette méthode se concentre sur les opérations affines.

La méthode repose sur la génération d'une série d'ellipsoïdes par l'application d'une matrice de transformation affine sur un sphéroïde unitaire. L'algorithme étire d'abord le sphéroïde à une taille suffisamment grande pour s'assurer d'englober au moins

une partie de l'espace solutions. Puis, tant que le centre de l'ellipsoïde viole une contrainte, on « coupe » l'ellipsoïde courant en son centre de façon parallèle à la contrainte identifiée. On modifie ensuite la transformation affine pour amener l'ellipsoïde suivant à englober le demi-ellipsoïde ainsi formé. Ce procédé est illustré à la figure 26. Éventuellement, on obtient un centre qui respecte toutes les contraintes du modèle ou l'ellipsoïde atteint une taille plus petite que la granularité de valeur de l'encodage binaire.

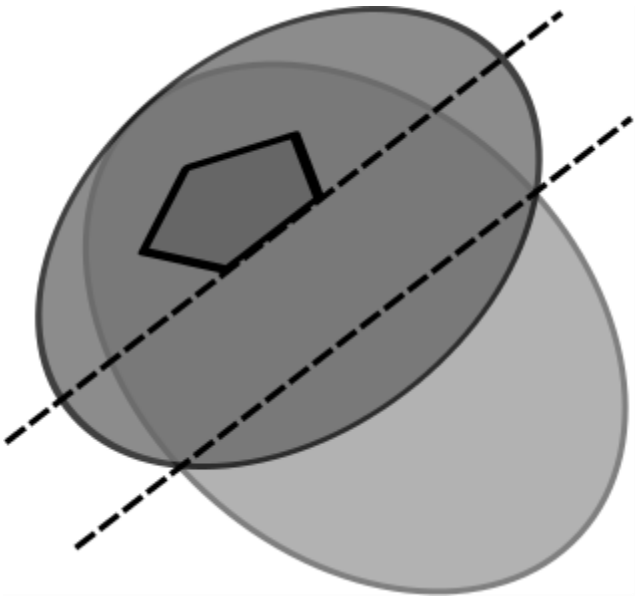

Figure 26 - Itération de la méthode des ellipsoïdes

Cet algorithme est théoriquement correct et se résout en temps polynomial dans son pire cas (ordre $O(n^6 \log_2 n)$). Cette performance est théoriquement meilleure que la méthode du simplexe. Cependant, la méthode demande une précision en fonction de la taille de l'instance, ce qui demande une stabilité numérique difficile à maintenir sur une machine discrète, discrétion essentielle à la terminaison de l'algorithme. De plus, l'algorithme simplexe demeure beaucoup plus efficace en pratique, ce qui lui limite l'avantage comparatif de la méthode des ellipsoïdes.

Algorithme de Karmarkar

L'algorithme de Karmarkar fut introduit en 1984 [26]. Il s'agit d'une forme d'approximation par convergence de Newton [27] ayant donné naissance à une famille d'algorithmes dits du point intérieur.

Cet algorithme repose sur l'idée de projeter l'espace solutions sur un hyperplan de façon à pouvoir en faire un simplexe « arrondi » autour d'un point sur lequel on peut obtenir une bonne évaluation. On déplace ensuite le point et un sphéroïde inscrit et on répète jusqu'à ce qu'on obtienne une solution optimale. Il suffit ensuite d'appliquer à rebours les projections effectuées pour replacer le point dans l'espace original.

Cet algorithme demande que le problème linéaire soit présenté sous une forme particulière, mais tout système d'inégalités linéaires peut être ramené sous cette forme. Un problème de taille mn sera de taille $(m+1)(n+2)$.

Cet algorithme est moins sensible à l'instabilité numérique que la méthode des ellipsoïdes et offre une complexité algorithmique en temps polynomial. Il est cependant lui aussi borné en fonction de la taille des valeurs du problème et sa complexité le rend moins accessible à la modification pour la navigation dans l'espace solutions.

Simulation Physique

Une approche basée sur la simulation physique a aussi été réalisée. L'idée est de poser les contraintes comme des collisions planes et d'effectuer un lancer de rayon en direction de la fonction d'optimisation en « glissant » sur les collisions jusqu'à atteindre un maximum.

Des algorithmes pour effectuer ce type de calcul existent dans le domaine de la simulation physique et du jeu vidéo. Cependant ces algorithmes se limitent à deux ou trois dimensions et reposent sur la connaissance de la topologie (sommets,

segments et faces) du polytope [24]. Ces informations peuvent être obtenues par l'algorithme NDND, mais les instances réelles sont problématiques étant donné leur taille.

Un algorithme a été envisagé pour appliquer le lancer de rayon sur des plans de contraintes formant un polytope convexe. S'il s'avérait efficace, la même approche aurait pu être utilisée pour la navigation dans l'espace solutions. L'algorithme a été baptisé gae bolga en référence au javelot du héros de la mythologie celte Cuchulain, javelot qui atteignait toujours sa cible et qui se multipliait à l'intérieur de celle-ci. Plus précisément, l'idée s'articule comme suit :

Soit P et un système d'inégalités linéaires S tel que $A_i x_i \leq b_i$ où l'on cherche à maximiser $c_i x_i$. Il faut tout d'abord transformer les inégalités linéaires en hyperplans $(A : \mathbb{R}^{m \times n}, b : \mathbb{R}^m) \mapsto (N : \mathbb{R}^n, d : \mathbb{R})^m$ et exprimer P et c sous la forme d'un rayon. Le corps de l'algorithme s'articule autour d'une boucle en deux temps : trouver la prochaine collision sur le chemin du rayon, puis ajuster la direction de celui-ci pour le faire « glisser » sur les collisions. Lorsque $G \cdot C \leq 0$, il n'est plus possible d'augmenter la valeur de C et le problème est borné. Lorsque l'algorithme ne trouve plus de collision, alors S est non-borné.

Le lancer de rayon trouve la distance δ entre le point Q et chacun des m hyperplans i avec l'équation $(Q + \delta G) \cdot N_i - d_i = 0$ ou sous forme équivalente $\delta = \frac{d_i - Q \cdot N_i}{G \cdot N}$ [24], une opération de la même complexité algorithmique que le pivot du simplexe. Cette approche a cependant l'avantage de ne pas modifier la matrice des données du modèle et est donc plus résistante face à l'instabilité numérique. De plus, on peut ignorer certains hyperplans par « *backface culling* » (lorsque $G \cdot N \geq 0$), ce qui allège les calculs.

Pour trouver le nouveau vecteur G lors de la collision à partir du point Q, on trouve un vecteur tel que pour tout hyperplan auquel appartient Q, $Q \cdot N \geq 0$ tout en minimisant $G \cdot C$. Conformément au lemme de Farkas [28], cette étape est en soit un problème linéaire, ce qui rend cet algorithme moins avantageux que le simplexe

qui, lui, repose sur une heuristique expérimentalement efficace. Afin de pouvoir utiliser cet algorithme, il faudrait trouver une heuristique pour déterminer G aussi efficacement que le simplexe choisit son pivot.

Pour conclure, rappelons que les systèmes à initiative partagée offrent un moyen intéressant pour la collaboration entre l'humain et l'ordinateur pour l'élaboration de solutions optimales aux systèmes linéaires. Nous avons proposé des améliorations aux systèmes existants afin de clarifier la présentation des données et de mieux intégrer les critères qualitatifs jugés par l'expérience et l'intuition humaines. Il reste cependant à élargir l'espace des solutions explorables pour couvrir la totalité de l'espace des solutions optimales.

Références

[1] S. Hamel, J. Gaudreault, C.-G. Quimper, M. Bouchard and P. Marier, "Human-Machine Interaction for Real-time Linear Optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, Séoul, 2012.

[2] M. A. Hearst, "Mixed-initiative interaction," *IEEE Inteligent Systems,* pp. 14-16, 1999.

[3] H. Karloff, Linear Programming, Boston: Springer Science & Business Media, 2008.

[4] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Mineola, New York: Dover Publications, inc., 1998.

[5] G. B. Dantzig, "Origins of the Simplex Method," Stanford Univesity, Stanford, 1987.

[6] S. S. Skiena, The Algorithm Design Manual, Londres: Springer, 2010.

[7] T. Yamada, J. Yoruzuya and D. Kataoka, "Enumerating extreme points of a highly degenerate polytope," *Computers Ops Research,* pp. 397-410, Avril 1994.

[8] J. R. Bunch, "The weak and strong stability of algorithms in numerical linear algebra," *Linear Algebra and Its Applications,* vol. 88, pp. 49-66, 1987.

[9] J. H. Wilkinson, Rounding Errors in Algebraic Processes, New York: Dover Publications, Inc., 1995.

[10] T. C. T. Kotiah and D. I. Steinberg, "Occurrences of cycling and other phenomena arising in a class of linear programming models," *Communications of the ACM,* pp. 107-112, 1977.

[11] A. M. Tavarez Thomé, L. F. Scavarda, N. Suclla Fernandez and A. J. Scavarda, "Sales and operations planning : A research synthesis," *International Journal of Production Economics,* pp. 1-13, 2012.

[12] M. Arabi, J. Gaudreault, M. Noureltfath, J. Favreau and M. Morneau-Pereira, "Integrating Optimization and Simulation for Supply Chain Tactical Planning in

the Forest Products Industry," in *4th International Conference on Information Systems, Logistics and Supply Chain CREATIVE LOGISTICS FOR AN UNCERTAIN WORLD*, Québec, 2012.

[13] H. P. Williams, Model Building in Mathematical Programming, 5e édition, Chichester: Wiley, 2013.

[14] W. Jerbi, J. Gaudreault, S. D'Amours, M. Nourelfath, S. Lemieux, P. Marier and M. Bouchard, "Optimization/simulation-based framework for the evaluation of supply chain management policies in the forest product industry," in *IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012)*, Séoul, Corée, 2012.

[15] G. Klau, N. Lesh, J. Marks and M. Mitzenmacher, "Human-Guided Search," *Journal of Heuristics, vol. 10,* pp. 289-310, 2010.

[16] M. Fleming, "The use of increasingly specific user models in the design of mixed-initiative systems," in *Proceedings of the 17th Conference of the Canadian Society for Computational Studies of Intelligence*, London, 2004.

[17] W. Kun and W. Havens, "Modeling an Academing Curriculum Plan as a Mixed-Iinitiative Constraint Satisfaction Problem," in *18th Conference of the Canadian Society for Computational Studies of Inteligence*, Victoria, 2005.

[18] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. G. Chafin, W. C. Dias and P. F. Maldague, "Mapgen: mixed-initiative planning and scheduling for the Mars exploration Rover mission," *IEEE Intelligent Systems 19,* pp. 8-12, 2004.

[19] J. L. Bresina and P. H. Morris, "Mission operations planning: Beyond MAPGEN," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology*, Pasadena, 2006.

[20] J. L. Bresina and P. H. Morris, "Mixed-initiative planning in space mission operations," *AI Magazine 28,* pp. 75-75, 2007.

[21] B. Guiost, S. Debernard, T. Poulain and P. Millot, "Supporting air-traffic controllers by delegating tasks," in *2004 IEEE International Conference on Systems, Man and Cybernetics.*, La Haye, 2004.

[22] T. Lenor, M. Lewis, S. Hahn, T. Payne and K. Sycarn, "Task characteristics and intelligent aiding [route-planning tasks]," in *2000 IEEE International Conference on Systems, Man and Cybernetics*, Nashville, 2000.

[23] I. L. Kantor, "Description of the Optimal Solution Set of the Linear Programming Problem and the Dimension Formula," *Linear Algebra and its Application,* pp. 19-32, 15 janvier 1993.

[24] C. Ericson, Real-Time Collision Detection, San Fransico: Elsevier, 2004.

[25] L. G. Khachiyan, "A polynomial Algorithm for Linear Programming," *Doklady Akademiia Nauk USSR 244,* pp. 1093-1096, 1979.

[26] N. Karmarkar, "A new Polynomial-Time Algorithm for Linear Programming," *Combinatorica,* pp. 373-395, 1984.

[27] P. E. Gill, W. Murray, M. A. Suaders, J. A. Tomlin and M. H. Wright, "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method," *Mathematical Programming,* pp. 193-209, 1986.

[28] J. Farkas, "Theorie der einfachen Ungleichungen," *Journal für die reine und angewandte Mathematik,* pp. 1-27, 1902.

[29] C. Williamson and B. Shneiderman, "The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System," *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval ,* pp. 338-346, 1992.

[30] E. R. Tufte, The Visual Display of Quantitative Information, Cheshire, Connecticut: Graphics Press, 1983.

Annexe 1 : Modèles linéaires utilisés pour évaluer les performances des algorithmes

Un jeu de six (6) modèles linéaires a été utilisé pour mesurer la validité et la performance des différents algorithmes proposés. Les modèles 1 et 2 sont des problèmes théoriques présentant une géométrie simple pour faciliter la visualisation de l'espace des solutions et l'espace des solutions optimales.

Modèle 1

(2 variables, 3 contraintes)

$$\begin{array}{ll} maximiser & x + y \\ s.o. & x + y \leq 10 \\ & -x + y \leq 3 \\ & x - y \leq 5 \\ & x, y \geq 0 \end{array}$$

Modèle 2

(3 variables, 6 contraintes)

$$\begin{array}{ll} maximiser & x + y + z \\ s.o. & x \leq 6 \\ & y \leq 6 \\ & x + y \leq 2 \\ & -x + y \leq 4 \\ & x + y + z \leq 10 \\ & x, y, z \geq 0 \end{array}$$

Modèle 3

(9 variables, 12 contraintes)

Le modèle trois est un modèle simple où la solution optimale est triviale afin de valider la rectitude des algorithmes. Une usine peut produire soit du papier ou des granules. La quantité de papier (le produit le plus payant) pouvant être produite est limitée à 126 unités par période pour une productivité périodique de 128 unités. On cherche à y maximiser le profit sur trois périodes.

$$
\begin{aligned}
&maximiser && profit && \\
&s.o. && p_i + g_i \leq 128 && \forall i \in [1,3] \\
& && p_i \leq 126 && \forall i \in [1,3] \\
& && p_T - \sum_{i=1}^{3} p_i = 0 && \\
& && g_T - \sum_{i=1}^{3} g_i = 0 && \\
& && -profit + 10000p_t + 1000g_t = 0 && \\
& && p_I, g_i, p_T, g_T, profit \geq 0 && \forall i \in [1,3]
\end{aligned}
$$

Modèle 4

(53 variables, 55 contraintes)

Le modèle 4 correspond à la planification d'heures de maintenance ou de repos dans une usine. Pour des semaines de production de 168 heures, on calcule un nombre d'heures totales inférieur pour l'année. La variante A du modèle considère l'équivalent d'une (1) semaine de repos et la variante B deux (2) semaines, Ce modèle permet de générer une grande quantité de points extrêmes dénombrables.

Variante A

$$
\begin{aligned}
maximiser \quad & p_T \\
s.o. \quad & p_i \leq 168 \qquad \forall i \in [1,52] \\
& p_T - \sum_{i=1}^{52} p_i = 0 \\
& p_T \leq 8568 \\
& p_i, p_T \geq 0 \qquad \forall i \in [1,52]
\end{aligned}
$$

Variante B

$$
\begin{aligned}
maximiser \quad & p_T \\
s.o. \quad & p_i \leq 168 \qquad \forall i \in [1,52] \\
& p_T - \sum_{i=1}^{52} p_i = 0 \\
& p_T \leq 8400 \\
& p_i, p_T \geq 0 \qquad \forall i \in [1,52]
\end{aligned}
$$

Modèle 5

(37 variables, 72 contraintes)

Le modèle 5 correspond à un réseau logistique simple où une forêt alimente un moulin à papier et une usine de granules. Les billots de bois récoltés sont acheminés par camion vers les sites de transformation. Le papier produit par le moulin est acheminé par camion pour être vendu à une papeterie et les granules produites par l'usine sont acheminées par train pour être vendues à une centrale à vapeur. Le schéma de ce réseau est présente à la figure 1.

Ce modèle présente des valeurs numériques plus près des ordres de grandeur des cas industriels réels. Il a été conçu pour valider la présentation des données dans l'interface utilisateur. C'est celui qui a été utilisé pour l'article faisant la publication des résultats.

$$\begin{aligned}
&maximiser && profit && \\
&s.o. && Coupe_i \leq 168 && \forall i \in [1,3] \\
& && Coupe_T \leq 504 && \\
& && Coupe_T - \sum_{i=1}^{3} Coupe_i = 0 && \\
& && ProductionPapier_i \leq 168 && \forall i \in [1,3] \\
& && ProductionPapier_T \leq 504 && \\
& && ProductionPapier_T - \sum_{i=1}^{3} ProductionPapier_i = 0 && \\
& && ProductionGranules_i \leq 168 && \forall i \in [1,3] \\
& && ProductionGranules_T \leq 504 && \\
& && ProductionGranules_T - \sum_{i=1}^{3} ProductionGranulles_i = 0 && \\
& && VentePapier_i \leq 10500 && \forall i \in [1,3] \\
& && VentePapier_T \leq 31500 && \\
& && VentePapier_T - \sum_{i=1}^{3} VentePapier_i = 0 && \\
& && VenteEnergie_i \leq 56000 && \forall i \in [1,3] \\
& && VenteEnergie_T \leq 168000 && \\
& && VenteEnergie_T - \sum_{i=1}^{3} VenteEnergie_i = 0 && \\
& && CamionForetMoulin_i \leq 1000000 && \forall i \in [1,3] \\
& && CamionForetMoulin_T \leq 3000000 &&
\end{aligned}$$

$$CamionForetMoulin_T - \sum_{i=1}^{3} CamionForetMoulin_i = 0$$

$$CamionForetUsine_i \leq 1000000 \qquad \forall i \in [1,3]$$

$$CamionForetUsine_T \leq 3000000$$

$$CamionForetUsine_T - \sum_{i=1}^{3} CamionForetUsine_i = 0$$

$$CamionMoulinPapeterie_i \leq 200000 \qquad \forall i \in [1,3]$$

$$CamionMoulinPapeterie_T \leq 600000$$

$$CamionMoulinPapeterie_T - \sum_{i=1}^{3} CamionMoulinPapeterie_i = 0$$

$$TrainUsineCentrale_i \leq 200000$$

$$TrainUsineCentrale_T \leq 600000$$

$$TrainUsineCentrale_T - \sum_{i=1}^{3} TrainUsineCentrale_i = 0$$

$$Coupe_i - 0.002CamionForetMoulin_i - 0.002CamionForetUsine_i = 0 \qquad \forall i \in [1,3]$$

$$Coupe_T - 0.002CamionForetMoulin_T - 0.002CamionForetUsine_T = 0$$

$$ProductionPapier_i - 0.002CamionForetMoulin_i = 0 \qquad \forall i \in [1,3]$$

$$ProductionPapier_T - 0.002CamionForetMoulin_T = 0$$

$$\frac{1}{6}CamionForetMoulin_i - CamionMoulinPapeterie_i = 0 \qquad \forall i \in [1,3]$$

$$\frac{1}{6}CamionForetMoulin_T - CamionMoulinPapeterie_T = 0$$

$$ProductionGranules_i - 0.002CamionForetUsine_i = 0 \qquad \forall i \in [1,3]$$

$$ProductionGranules_T - 0.002CamionForetUsine_T = 0$$

$$\frac{2}{3}CamionForetUsine_i - TrainUsineCentrale_i = 0 \qquad \forall i \in [1,3]$$

$$\frac{2}{3}CamionForetUsine_T - TrainUsineCentrale_T = 0$$

$$CamionMoulinPapeterie_i - VentePapier_i = 0 \qquad \forall i \in [1,3]$$

$$CamionMoulinPapeterie_T - VentePapier_T = 0$$

$$TrainUsineCentrale_i - VenteEnergie_i = 0 \qquad \forall i \in [1,3]$$

$$TrainUsineCentrale_T - VenteEnergie_T = 0$$

$$profit - 120VentePapier_T - 30VenteEnergie_T = 0$$

Modèle 6

(42 variables, 111 contraintes)

Le modèle 6 représente le réseau de création de valeur simple présenté à la section 1.5. Réseau de création de valeur.

$$\text{maximiser} \quad \sum_{1=1}^{3} (2VentePapier_i + 2VenteEnergie_i - AchatEnergie_i)$$

$$\begin{aligned}
s.o. \quad & Coupe_i \leq 10 && \forall i \in [1,3] \\
& AchatEnergie_i \leq 3 && \forall i \in [1,3] \\
& ProductionPapier_i \leq 10 && \forall i \in [1,3] \\
& ProductionGranulles_i \leq 10 && \forall i \in [1,3] \\
& ProducitonEnergie_i \leq 10 && \forall i \in [1,3] \\
& VentePapier_i \geq 2 && \forall i \in [1,3] \\
& VenteEnergie \leq 5 && \forall i \in [1,3] \\
& VenteEnergie_i \geq 3.5 && \forall i \in [1,3] \\
& CamionForetMoulin_i \leq 10 && \forall i \in [1,3] \\
& CamionForetUsine_i \leq 10 && \forall i \in [1,3] \\
& LigneFournisseurMoulin_i \leq 10 && \forall i \in [1,3] \\
& TrainUsineCentrale_i \leq 10 && \forall i \in [1,3] \\
& LigneCentraleMoulin_i \leq 10 && \forall i \in [1,3] \\
& LigneCentraleHydro_i \leq 10 && \forall i \in [1,3] \\
& CamionMoulinPapetrie_i \leq 10 && \forall i \in [1,3] \\
& CamionForetMoulin_i + CamionForetUsine_i - coupe_i = 0 && \forall i \in [1,3] \\
& LigneFournisseurMoulin_i - AchatEnergie_i = 0 && \forall i \in [1,3] \\
& CamionMoulinPapetrie_i - ProductionPapier_i = 0 && \forall i \in [1,3] \\
& CamionForetMoulin_i - ProductionPapier_i = 0 && \forall i \in [1,3] \\
& LigneFournisseurMoulin_i + LigneCentraleMoulin_i - CamionMoulinPapetrie_i = 0 && \forall i \in [1,3] \\
& CamionForetUsine_i - ProductionGranulle_i = 0 && \forall i \in [1,3] \\
& CamionForetUsine_i - TrainUsineCentrale_i = 0 && \forall i \in [1,3] \\
& TrainUsineCentrale_i - ProductionEnergie_i = 0 && \forall i \in [1,3] \\
& LigneCentraleMoulin_i + LigneCentraleHydro_i - TrainUsineCentrale_i = 0 && \forall i \in [1,3] \\
& CamionMoulinPapetrie_i - VentePapier_i = 0 && \forall i \in [1,3] \\
& LigneCentraleHydro_i - VenteEnergie_i = 0 && \forall i \in [1,3] \\
& coupe_i, AchatEnergie_i, ProductionGranulles_i, ProductionEnergie_i, CamionForetMoulin_i, CamionForetUsine_i, LigneFournisseurMoulin_i, TrainUsineCentrale_i, LigneCentraleMoulin_i, LigneCentraleHydro_i, CamionMoulinPapetrie_i \geq 0 && \forall i \in [1,3]
\end{aligned}$$

Annexe 2 : Article traitant des avancements effectués

Les résultats obtenus pour l'ajout d'une interface personne-machine et de l'ajout d'une tolérance à l'optimalité ont été publiés dans les actes d'une conférence avec comité de lecture (« A mixed-initiative system for interactive tactical supply chain optimization », *10ème Conférence Francophone de Modélisation, Optimisation et Simulation : de l'économie linéaire à l'économie circulaire*, 5-7 novembre 2014). J'en suis l'auteur principal et j'ai effectué toutes les expérimentations qui y sont présentées. Les coauteurs sont mon directeur et mon codirecteur de recherche qui m'ont dirigé dans mes travaux.

10th International Conference of Modeling and Simulation - MOSIM'14 – November 5-7 - Nancy - France "Toward circular Economy"

A MIXED-INITIATIVE SYSTEM FOR INTERACTIVE TACTICAL SUPPLY-CHAIN OPTIMIZATION

François Chéné, Jonathan Gaudreault, Claude-Guy Quimper

FORAC Research Consortium – Université Laval – Québec, Canada, G1V 0A6

Corresponding author: jonathan.gaudreault@forac.ulaval.ca

ABSTRACT: Tactical supply-chain planning consists of deciding the amount of products produced, consumed, stored or transported for each period of a given time frame with a precise objective to optimize. Many mathematical models have been proposed but most of the time they do not exploit decision makers intuition and preferences. MixedInitiative Systems (MIS) aim to solve problems in a different way, coordinating the interaction of intelligent automated agents and humans. MIS were developed for discrete combinatorial optimization but recently, a novel MIS approach has been proposed for problems showing a linear structure, like tactical supply-chain optimization. This approach allows a decision maker to explore a space of optimal solutions. In this paper, this scheme is improved in order to allow a user to explore near-optimal solutions as well. Furthermore, instead of interacting with lowly evocative bar charts, the decision maker uses the strongly visual map-based LogiLab user interface which allows him to view/modify in real time the flow between businesses of a forest-products supply chain.

KEYWORDS: Forest-products supply chain; Tactical planning; Mixed-Initiative Systems; Linear optimization

1 INTRODUCTION

Operation planning requires profound knowledge, solid experience, acute instinct, and sufficient information. While computers excel in treating large amounts of information, they often lack the knowledge and experience of human managers (Klau *et al.*, 2010). This has led some researchers to propose *Mixed-Initiative Systems* (MIS) that promote the interaction of a human decision maker with an automated computing agent. (Allen, 1999; Hearst, 1999)

Classical Mixed-Initiative Systems were developed for discrete combinatorial optimization. Recently, Hamel *et al.* (2012) proposed a MIS approach (Hamel's method) for linear problems like supply chain tactical optimization. Within Hamel's approach, the system precomputes a set of optimal solutions and the user can explore the solution space by interacting with bar charts showing the current solution. Following each variable modification requested by the user, the system recomputes and displays in real time a new optimal solution.

In this paper, this scheme is improved in order to allow the user to explore near-optimal solutions as well. Furthermore, instead of interacting with bar charts, the decision maker uses the map-based LogiLab user interface which allows him to view and modify in real time the flow between businesses of a forest-products supply chain. The user also has the option of editing solutions using

aggregated results (e.g. annual flows), or detailed results (e.g. flows per period).

The remainder of the paper is organized as follows. Section 2 presents preliminary notions on supply chain optimization, Mixed-Initiative Systems (MIS), and Hamel's method for interactive flow optimization. Section 3 presents the LogiLab forest-products tactical supply-chain optimization system. We propose some modifications to its graphical user interface that allow using it as an MIS exploiting Hamel's method. Then, in Section 4, we expand Hamel's methods in order to allow the user to explore an expended set of (near-optimal) solutions. Section 6 presents a proof of concepts together with experimental results for a forest-products supply chain. Section 7 concludes the paper.

2 PRELIMINARY NOTIONS

2.1 Supply Chain Optimization (SCO)

Supply chains are formed by a set of business units where processes consume resources to generate products. Resources can be local to the business units, like the hours that can be worked, or products of other processes, which can occur in other business units. They also include external resource suppliers and clients. Generally, products must be transported between business units.

Tactical supply-chain planning consists of computing the amount of products produced, consumed, stored or transported for each period of a given time frame with a precise objective to optimize. Common objectives are costs minimization or profit maximization.

Such problems can easily be modeled using a linear programming model. They can be solved using wellknown algorithms, like the Dantzig (1955) simplexe algorithm.

Tactical supply-chain planning is a very important concern in the forest-products industry. This industry is facing divergent product flows (Haartveit *et al.*, 2004). A single business unit produces many products at the same time from a single piece of raw material. This leads to an important interdependency between business units (e.g. forest operations supply many different types of industries at the same time, a sawmill supplies lumber to remanufacturing plants as well as chips to paper mills, etc.). Many mathematical models have been proposed (Jerbi *et al.*, 2012; Singer and Donoso, 2007) but most of the time they do not exploit decision makers' intuition and preferences.

2.2 Mixed-Initiative Systems (MIS)

In order to complete a complex task, one may gather multiple specialists of different trades relevant to the problem at hand. It is quite difficult to model preferences of many decision makers. They are often unaware of their own preferences before seeing a solution that violates them. Mixed-initiative systems aim to solve problems in a different way, coordinating the interaction of intelligent automated agents and humans (Hearst, 1999).

Involving humans in the search for an optimal solution provides multiple benefits. Humans generally outperform computers in visual perception and strategic thinking. They can also justify and improve solutions in which they participate, and they can implement their preferences and knowledge of the real world without complex and sometimes near-impossible mathematical modeling. Finally, human participation generates a stronger trust in the produced solution (Klau *et al.*, 2010).

Most MIS-related research targets discrete combinatorial optimization problems such as timetabling (Kun and Havens, 2005), space mission planning and scheduling (Ai-Chang *et al.*, 2004; Bresina and Morris, 2006, 2007), air traffic control (Guiost *et al.*, 2004), military applications (Lenor *et al.*, 2000; Linegang *et al.*, 2003), etc.

2.3 MIS for linear optimization

Hamel *et al.* (2012) proposed an MIS approach for linear optimization. The system allows the user to visualize an implicit sub-space of optimal solutions for a given set of variables. The user can interactively increase or decrease the value of a variable and the system reacts by computing and displaying, in real time, the new subspace of optimal solutions (Figure 1).

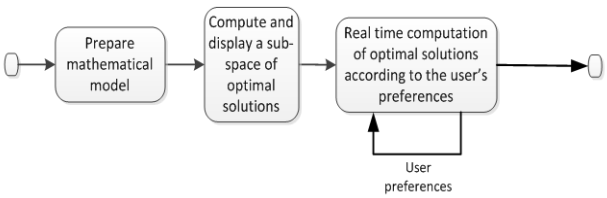

Figure 1. MIS concept applied to linear programming

(Adapted from Hamel *et al.* 2012)

The system works as follows. It uses a classical linear programming solver to compute an optimal solution to a problem *P*. Once the optimal objective value is known, it searches for other optimal solutions by creating a new instance of the problem, called *P'*, by adding a constraint forcing the objective value to be equal to the optimal value found for *P*. The space of *feasible* solutions of *P'* is therefore equivalent to the space of *optimal* solutions of *P*. For each of the *n* variables x_i displayed to the user, the system searches for a solution $\underline{s_i}$ that minimizes x_i and a solution $\overline{s_i}$ that maximizes x_i while preserving the optimality of the solutions. These 2*n* computations can be performed in parallel using a supercomputer. Then, one can display on the chart (see

Figure 2) the *range of optimality* for each variable x_i (the minimum and maximum values the variable can take in an optimal solution).

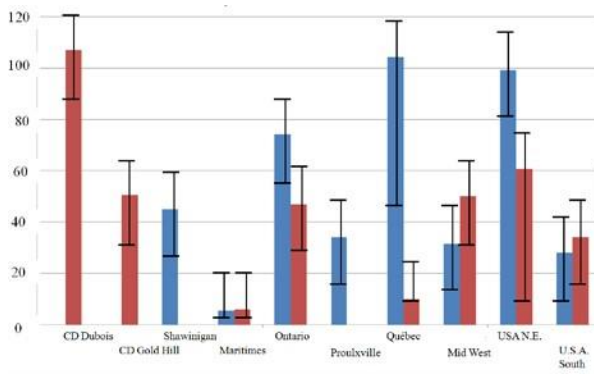

Figure 2. An optimal solution showing the range of optimality for the main decision variables

A chart like the one depicted in Figure 2 allows the user to see the flexibility (range of optimality) he has for any of the variables of interest. He can thus change the value of a variable within the optimality zone of that variable, for example by dragging up or down the top of a bar in the chart. Upon the modification of a variable, the system adjusts the value taken by the other variables such that the solution remains feasible and optimal. Normally, this operation would require a lengthy complete re-optimization of the problem. However, we know that it is possible to compute a new optimal solution by generating a convex combination of the optimal solutions obtained from the previous section. To find a solution with a given variable assigned to a specific value, it is sufficient to compute a new convex combination of extreme solutions.

This can be done in real time and it allows instantly refreshing the chart and displaying the impact of the user modification on the other variables. The user can thus *navigate* through the solution space, successively modifying several variables until a suitable solution is found.

Figure 3 illustrates this idea for a small problem with two variables. The gray polygon represents the set of convex combinations formed by optimal solutions minimizing and optimizing each of the variables[1].

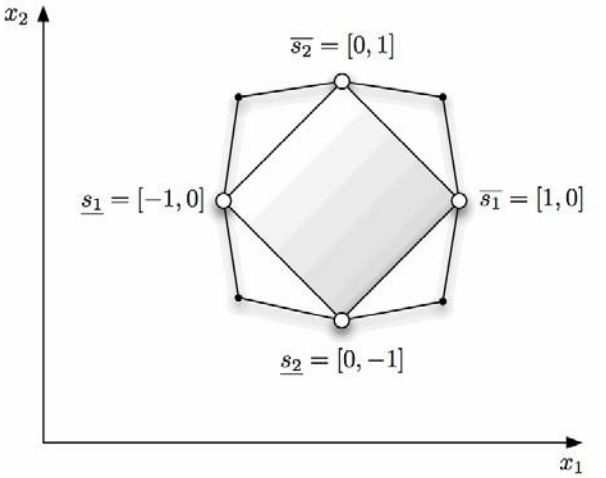

Figure 3. Available optimal zone using convex combinations

When the user modifies a variable, the system not only looks at any point in the sub-space such that the modified variable takes the desired value, but it also looks for a solution such that the other variables move as little as possible so that the overall system appears to be *stable*. Given a solution *x*, a variable x_i, and a value *v*, finding a

[1] This example (Figure 3) also shows that there could be optimal solutions (in white) not covered by this sub-space (in gray). This is why we say that the user navigates in a space (formed by convex combination of our extreme solutions) that is in reality a sub-space of the optimal solutions.

new solution x' where $x_i' = v \neq x_i$ while changing as little as possible the values of the other variables is an optimization problem.

Since we aim to have the human agent to interact in realtime with the system, responsiveness was an important criterion, Hamel's method looks for a solution in the convex hull of the 2*n* precomputed solutions $s_1, s_1, \ldots, s_n, s_n$, a much smaller problem than the original one. Hamel proposed what he called the *triangular heuristic*. First, it determines if the user asked for the variable x_i to be increased ($v > x_i$) or decreased ($v < x_i$). If the variable is increased, he computes the unique convex combination between the current solution *x* and $\overline{s_i}$ that intersects the hyperplane $x_i = v$. If the variable is decreased, he computes the unique convex combination between the current solution *x* and s_i that intersects the plane $x_i = v$. In other words, if the decision maker increases x_i, he sets

$$x' = \alpha x + (1 - \alpha)\overline{s_i}$$

where

$$\alpha = \frac{\overline{s_{ii}} - v}{\overline{s_{ii}} - x_i.}$$

If the decider decreases the variable x_i, he sets

$$x' = \alpha x + (1 - \alpha)s_i$$

where

$$\alpha = \frac{s_{ii} - v}{s_{ii} - x_i}$$

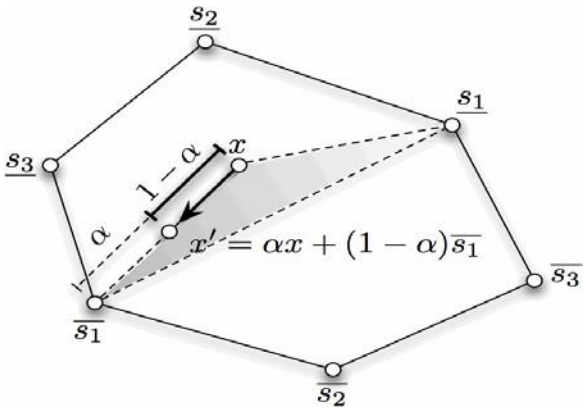

Figure 4. An example of the triangular method

In the next sections, we explain how Hamel's method can be adapted to allow the user to explore an expanded set of (near-optimal) solutions. We also show how we adapted the user interface in order to allow the user to interact with flows on a map instead of bar charts.

3 PROPOSED GRAPHICAL USER INTERFACE (GUI) FOR INTERACTIVE SUPPLY CHAIN TACTICAL OPTIMIZATION

The FORAC Research Consortium developed a software named LogiLab to optimize and display the design and planning of forest-products supply chains. The mathematical optimization model used by LogiLab is described by Jerbi *et al.* (2012).

Since LogiLab's data presentation is strongly evocative and natural for planners (See figure 5), we decided to use it as a basis for our interactive system. In the original LogiLab, the products flows from one business unit to another are presented by arcs connecting the corresponding nodes. The thickness of each arc is proportional to the volume of products that flows from its origin to its destination.

As we now want to display the "range of optimality" of each individual variable, we proposed replacing each arc by a "pipe". According to this metaphor, the diameter of the pipe represents the maximum value the flow variable can take. The colored part of the pipe represents the current level of goods that flows through the pipe (i.e. value of the variable in the current optimal solution). To represent the minimal value that allows for the solution to be optimal, the bottom part of the flow in the pipe is displayed with a darker color (see Figure 6).

The simplest and most intuitive way to allow the user to interact with the flows (in order for him to adjust the value of a variable) is simply to allow him to click on the pipe and then drag the mouse within the range of optimality. However, most flows tend to be too small to enable proper manipulation. Therefore, when the user rolls the cursor over a flow, an enlarged cross-section appears and allows easier manipulations (see Figure 4).

In the original LogiLab, the percentage of the total capacity of a business unit used is displayed using a rectangle similar to a progression bar

(look at the colored rectangle under each icon in Figure 5).

We added visualization of the range of optimality data for these variables using a color code similar to what we proposed for the flow variables.

Figure 6 presents the graphical user interface integrating those concepts. It presents an interaction scenario: (1) The user rolls over a flow; the zoom bubble (that resembles the cross-section of a pipe) appears and the user is allowed (2) to set a new value for this flow variable. Finally (3), the system updates in real time the value of the other variables, thus computing a feasible solution that is still optimal.

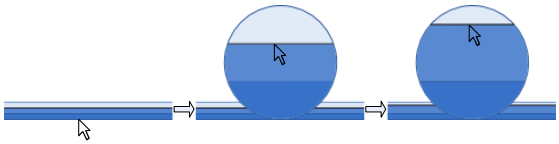

Figure 4. Interaction with flows within the proposed system

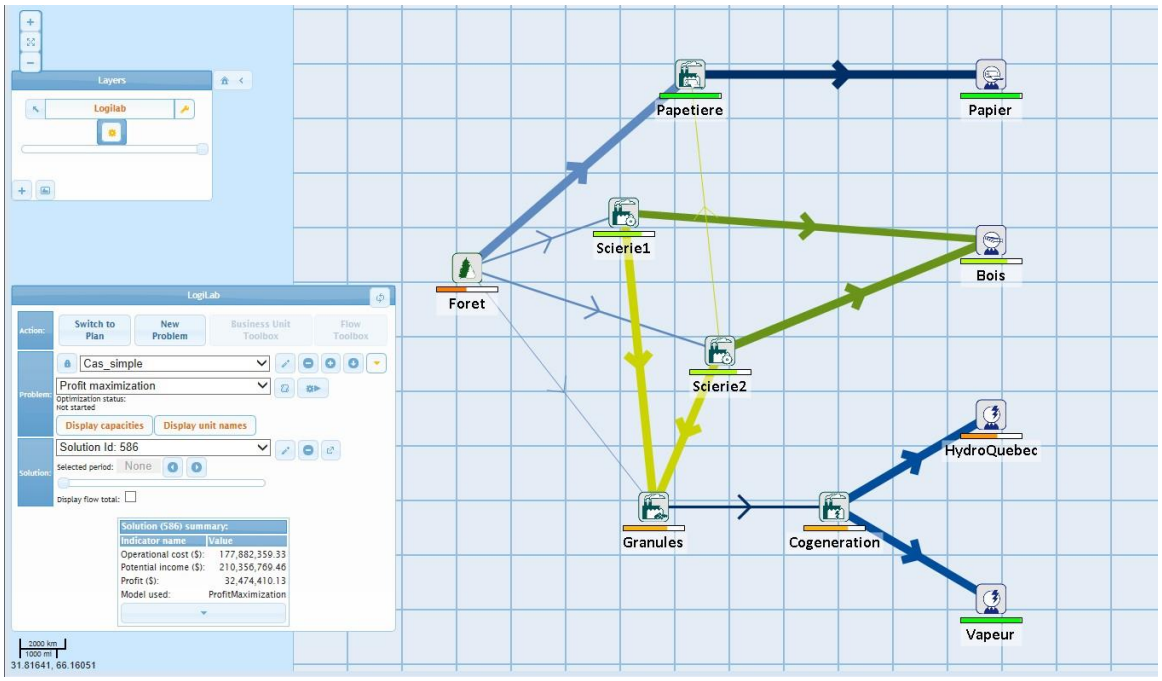

Figure 5. LogiLab graphical user interface

(1)

(2)

(3)

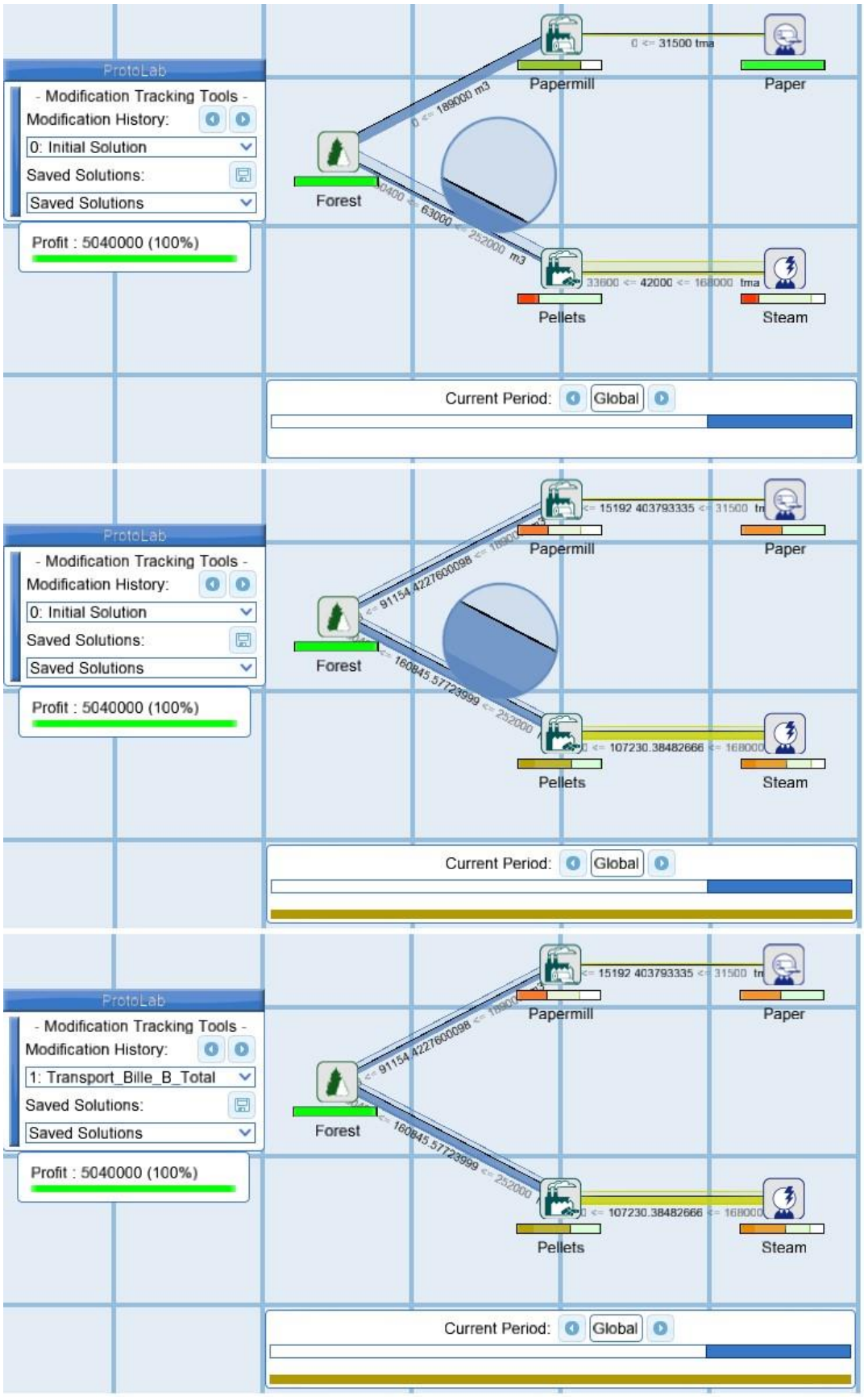

Figure 6. Proposed graphical user interface for interactive supply chain tactical optimization

4 INTRODUCING OPTIMALITY TOLERANCE (GAP) INTO HAMEL'S METHOD

Hamel's original method, even coupled to our new graphical user interface, only allows exploring optimal solutions. However, decision makers may be interested in exploring near-optimal solutions too (as the data used to model the problem are often rough estimates, it would be pointless to exclude solutions that decrease profit by only a few dollars).

The first (naive) approach we tried to modify Hamel's approach was the following. When computing solutions minimizing/maximizing a given variable, we tolerate a gap between the new solution's profit and the optimal solution of the original problem.

However, this approach does not work very well when using the triangular heuristic. Interaction with the system revealed frequent cases where adjusting a variable to values within the optimal space would return a nearoptimal solution while some existing optimal solutions would clearly be a better choice. We aim to explain the cause of this behavior and to propose another approach.

With the triangular heuristic, as the values of the variables are interpolated between two solutions, so are the profit values associated to these solutions. Therefore, if one solution is optimal but not the other, unless the weight of the optimal solution is 0 and the weight of the non-optimal solution is 1, then the solution resulting from the combination of these solutions will not be optimal.

This was not an issue with Hamel's original method where all solutions were optimal. However, with the modified approach, we combine non-optimal solutions. This result is that the system (1) returns a non-optimal solution as soon as a variable is modified towards an extreme value associated with a non-optimal solution even if the target value is within the optimal range for that variable (see Figure 7) and (2) once a non-optimal solution is reached, the system does not return an optimal solution unless a variable is set to an extreme value associated with an optimal solution.

This behavior is explained in Figure 7. In this example, the green area is the real optimal solution space. The yellow area shows how this space expands when we tolerate an optimality gap. In this example, we suppose the system first displays the solution for which y is maximized and then the user asks to gradually decrease x . As soon as x starts to decrease, the triangular heuristic gets a sub-optimal solution as it interpolates between y and $\underline{\underline{x}}$. This is a pity as it is clear that for some values for x we could still have an optimal (green) solution.

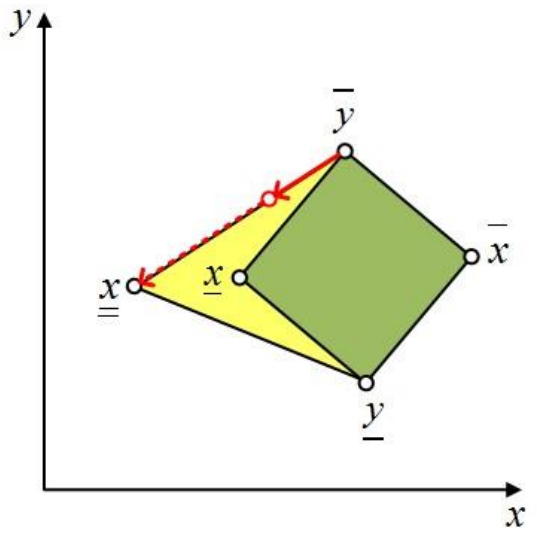

Figure 7. Behavior of Hamel's original Triangular heuristic when provided with a solution space with optimality gap

The situation is even worse for the following reason. Suppose we use a solver (e.g. the simplexe method) to find a solution minimizing/maximizing a given variable, with the constraint that the profit of the network should be at least 95% of the profit of the original optimal solution. The simplexe could return a solution minimizing/maximizing the variable that has a profit of 95% even if there exists another solution where the variable takes the exact same value while showing a profit of 100%.

To palliate this situation, we propose a multi-stage approach in the next subsection.

4.1.1 Proposed multi-stage approach

We want the system to return an optimal solution whenever possible. To achieve this, we need two pairs of solutions for each variable x_i the user is interested to modify. As in Hamel's original

method, one pair of solutions $\{x\ x_i\ ,\ _i\}$ minimizing/maximizing the variable value while preserving optimality and another pair $\{x\ x_i\ ,\ _i\}$ minimizing/maximizing the variable value

obtained when the optimality constraint is relaxed (according to some gap specified by the user).

We also modified the triangular heuristic to interpolate

$=$

using tolerated extreme points x_i and x_i only when

$=$

necessary.

This behavior is illustrated in Figure 8. In this example, we suppose the system first displays the solution for which y is maximized. Then the user asks to gradually decrease x . As long as x is greater than the value it has in solution $\underline{x}$, we interpolate between the solutions y and $\underline{x}$ and the solution is still optimal. Past that point, we interpolate between the solutions $\underline{x}$ and $\underline{\underline{x}}$.

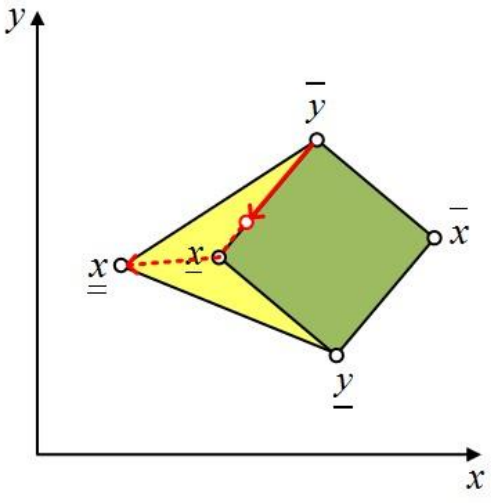

Figure 8. Behavior of the multi-stage approach

5 PROOF OF CONCEPT

In this section, the proposed multi-stage approach is compared to the original Hamel method and to the Hamel method with tolerance.

We consider a small linear optimization problem (wood flow supply chain) that we model with LogiLab. This problem is graphicly represented in Figure 6. Logs are harvested from a forest and can either be transformed into paper or pellets. Both products cost the same to produce and generate the same income per log. Paper demand is below production capacity, implying that a minimal amount of pellets must be produced to maximize the profit. We suppose a planning horizon of 3 periods and we have a total of 37 variables.

Using the *Hamel's original approach*, the range of optimality of the variables is rather small.

The decision maker now wants to interactively explore/discover near optimal solutions (max gap. 5 %) that best meet his preferences. Introducing tolerance should allow decreasing pellet production, and/or reduce forest harvesting volumes.

Using the *Hamel with gap tolerance* approach, variables should provide a wider range of optimality (thanks to the allowed 5 % gap) but most manipulation of a variable by the user should lead to non-optimal solutions (even though variables are manipulated within their range of optimality).

With the *multi-stage approach*, we should have the same range of optimality for the variables, but a small modification to these variables should lead to optimal solutions.

In the following experiment, we formally compare the approaches according to these criteria: (1) range of optimality of the variables, and (2) the optimality of the solutions that are computed after a modification to a variable.

For each method, we computed the range of optimality of each variable. Then, for each variable, we incremented/decremented the variable and measured the gap between the new solution and the original one.

Figure 9 shows how the range of optimality of individual variables are extended (in comparison to the original Hamel method) when we allow a 5 % gap (results are the same for Hamel with gap

tolerance and the multi-state approaches). We see no increase for 16 variables, a 5% increase for 6 variables (including of course the objective-function "profit" variable) and a 15% increase for 15 other variables.

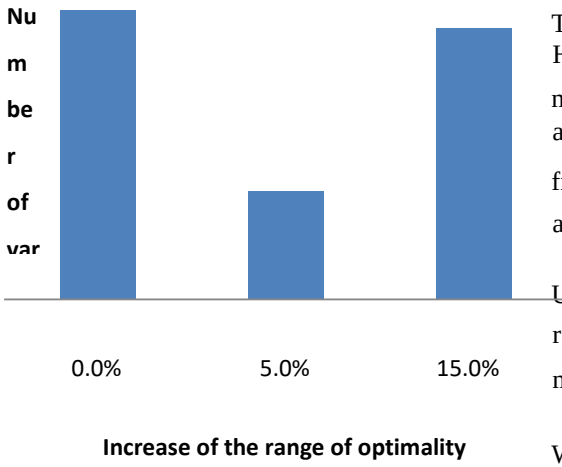

Figure 9. Increase of the range of optimality of the individual variables when provided with a 5% gap on the solution objective function

The ill effects of the *Hamel with tolerance gap* approach appeared as expected. Out of the 72 "extreme" solutions, ($\underline{\underline{x}}$ and $\bar{x}$ minimizing/maximizing the 36 manipulable variables), 52 were non-optimal.

Except for the pellet production and the forest harvesting variables, even small modification to a variable leads to a sub-optimal solution (in which pellet production and forest harvesting decrease). Consequently, the system gives the users the false impression they cannot modify these variables without altering optimality.

Using the *multi-stage* approach, out of the 144 solutions corrresponding to $\underline{x}, \ldots, \bar{x}$ $\underline{\underline{x}}$ $\bar{x}$, only 20 were sub-optimal (they correspond to the situations where $\underline{\underline{x}} < \underline{x}$ or $\bar{x} > \bar{x}$). All other manipulations of a variable in the range between $\underline{x}$ and $\bar{x}$ lead to optimal solution. This can be verified for any variable but we report results for one specific variable in Figure 10. For the purpose of our demonstration, we selected a variable that has some range of optimality even when the optimality gap is zero.

The blue curve shows that using the original Hamel method, we can modify the variable from $\bar{x}$ to $\underline{x}$ and from $\underline{x}$ to $\bar{x}$ and the new computed solution is always optimal.

Using the Hamel method with gap tolerance (in red) we leave the optimality region as soon as we move from $\underline{x}$ toward $\underline{\underline{x}}$.

With the multi-stage approach (in green), the solution is optimal between $\bar{x}$ and $\underline{\underline{x}}$. We leave optimality only if we go below $\underline{x}$ toward $\underline{\underline{x}}$, which is the expected behavior. A similar behavior is shown when we move back to $\bar{x}$.

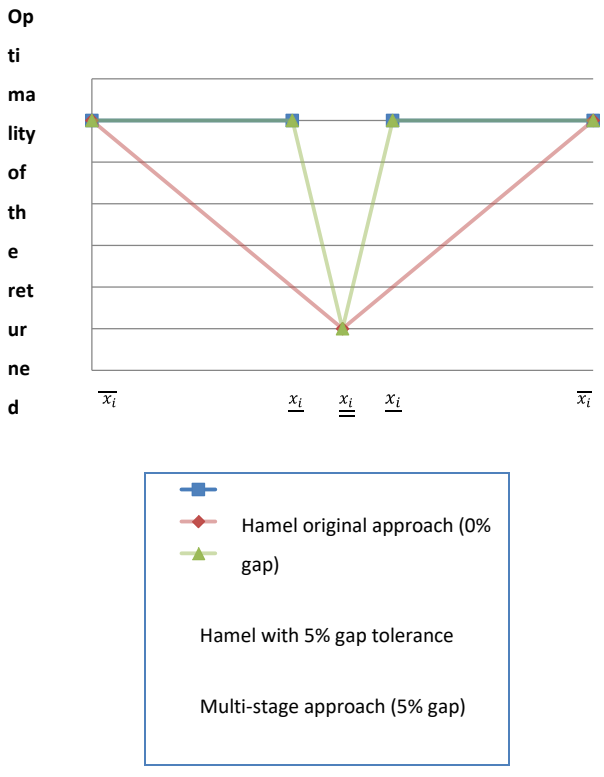

Figure 10. Comparison of optimality of returned solutions during interaction. For this variable, the

– =

= solutions for x and x were identical, this is why x does not appear in the chart.

6 CONCLUSION

Interactive linear optimization with human implication in a Mixed-Initiative System offers promising possibilities. Previous research has produced interesting results for the linear optimization problem. We deemed it appropriate to pursue this research through improvement of the mechanism by introducing optimality tolerance.

We proposed a new user interface for interactive supply chain optimization based on the LogiLab graphical user interface, a software developed for supply chain optimization in the forest-products industry.

We presented an experimentation that illustrates how the user experience of the decision makers and the solution space he can explore in real time can be expanded using the proposed approaches.

REFERENCES

Ai-Chang, M., Bresina, J., Charest, L., Chase, A., Hsu,

J.C.J., Jonsson, A., Kanefsky, B., Morris, P.,

Kanna,

R., Yglesias, J., Chafin, B.G., Dias, W.C., Maldague, P.F., 2004. MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. IEEE Intelligent Systems 19, 8-12.

Allen, J.E., 1999. Mixed-initiative interaction. IEEE Intelligent Systems 14, 14-16.

Bresina, J.L., Morris, P.H., 2006. Mission operations planning: beyond MAPGEN, Second IEEE International Conference on Space Mission Challenges for Information Technology. IEEE Comput. Soc, Pasadena, California, pp. 477-484.

Bresina, J.L., Morris, P.H., 2007. Mixed-initiative planning in space mission operations. AI Magazine 28, 75-88.

Dantzig, G.B., 1955. Linear programming under uncertainty. Management science, 197-206.

Guiost, B., Debernard, S., Poulain, T., Millot, P., 2004. Supporting air-traffic controllers by delegating tasks, Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics. IEEE, The Hague , Netherlands, pp. 164-169.

Haartveit, E.Y., Kozak, R.A., Maness, T.C., 2004. Supply chain management mapping for the forest products industry: Three cases from western Canada. Journal of Forest Products Business Research 1, 31.

Hamel, S., Gaudreault, J., Quimper, C.-G., Bouchard, M., Marier, P., 2012. Human-machine interaction for real-time linear optimization, 2012 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012), Seoul, Korea, pp.

673-680.

Hearst, M.A., 1999. Mixed-Initiative Interaction. IEEE Intelligent Systems 14, 14.

Jerbi, W., Gaudreault, J., D'Amours, S., Nourelfath, M., Lemieux, S., Marier, P., Bouchard, M., 2012. Optimization/simulation-based framework for the evaluation of supply chain management policies in the forest product industry, IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012), Seoul, Korea, pp. 1742-1748.

Klau, G., Lesh, N., Marks, J., Mitzenmacher, M., 2010. Human-guided search. Journal of Heuristics 16, 289310.

Kun, W., Havens, W.S., 2005. Modelling an academic curriculum plan as a mixed-initiative constraint satisfaction problem, Proceedings of the 18th Conference of the Canadian Society for Computational Studies of Intelligence (LNCS #3501). Springer-Verlag, Victoria, Canada, pp. 7990.

Lenor, T., Lewis, M., Hahn, S., Payne, T., Sycarn, K., 2000. Task characteristics and intelligent aiding, Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics. IEEE, Piscataway, NJ, USA, pp. 1123-1127.

Linegang, M., Haimson, C., MacMillan, J., Freeman, J., 2003. Human control in Mixed-Initiative Systems: lessons from the MICA-SHARC program, Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics. IEEE, Washington, D.C., pp. 436-441.

Singer, M., Donoso, P., 2007. Internal supply chain management in the Chilean sawmill industry. International Journal of Operations & Production Management 27, 524-541.