Human-Machine Interaction for Real-time Linear Optimization

Simon Hamel, Jonathan Gaudreault*, Claude-Guy Quimper, Mathieu Bouchard, Philippe Marier
FORAC Research Consortium, Université Laval
Québec, Canada, G1V 0A6

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

Abstract — **Mixed-Initiative-Systems (MIS) are hybrid decision-making systems in which human and machine collaborate in order to produce a solution. This paper described an MIS system adapted to business optimization problems. These problems can be solved in less than an hour as they show a linear structure. However, this delay is unacceptable for iterative and interactive decision-making contexts where users need to provide their input. Therefore, we propose a system providing the decision-makers with a convex hull of optimal solutions minimizing/maximizing the variables of interest. The users can interactively modify the value of a variable and the system is able to recompute a new optimal solution in a few milliseconds. Four real-time reoptimization methods are described and evaluated.**

Keywords — Linear optimization; Mixed-initiative systems; Supply chain optimization; Human-machine interaction.

I. Introduction

Most decision-making systems (e.g. planning or scheduling systems) found in enterprises lie on one of the following paradigms. The first one is fully automated systems. It is typically the case when an algorithm is used to find an optimal solution to the decision problem. In other cases, the planning is done by a human expert, sometimes with the help of a visual interface allowing him to get real time feedback regarding his decisions and choices. Surprisingly, quite a few optimization problems are planned manually as such. Indeed, automated planning tools are lacking of a political sensitivity or, more generally, do not take into consideration many important soft constraints that are often quite difficult to model (constraints that even human does not realize they exists before he sees a solution violating them).

Mixed-Initiative-Systems (MIS) [1, 2] are hybrid decision-making systems in which human and machine collaborate in order to produce a solution. Most MIS-related research is done by the A.I. community and applies to discrete combinatorial optimization problems.

The goal of this research is to propose MIS methods adapted to business optimization problems showing a linear structure. Preliminary notions regarding MIS and supply chain optimization are provided in Section II. The proposed Mixed-Initiative system for linear optimization is described in Section III. It is then evaluated for a real-size industrial supply chain problem in Section IV. Finally, section V concludes the paper.

II. PRELIMINARY NOTIONS

A. *Mixed-Initiative Systems (MIS)*

The motivation behind MIS is that human and machines show different strength [3]. Human has an implicit knowledge of the problem that cannot always be formalized. Human guidance can improve the performance of search algorithms [4]. Moreover, the decision-maker is often unaware of a constraint or an issue till he sees it in the solution proposed by the machine. In this context, involving the human in the search for the solution has some values.

Depending on the context, MIS provide different benefits. In some situations, the solution is produced using less computation time because the user can guide the search according to his intuition [4]. In other contexts, the main interest is for the final solution to get a better acceptance level by decision-makers because it is more in line with informal objectives of the company/decision maker.

Most MIS-related researches target discrete combinatorial optimization problems such as timetabling [5], space mission planning and scheduling [6-8], air traffic control [9], military applications [10, 11], etc.

Mixed-Initiative Systems have not yet made their way into business management systems such as those used for Supply Chain Management (SCM). We believe the main reasons are that (1) most of these situations can be modeled as linear optimization problems for which good algorithms exists, and (2) methods developed for classical A.I. planning could not easily be applied to those kind of problems.

B. *Products flow optimization in supply chain management*

The goal of our research is to propose MIS methods adapted to business problems showing a linear structure model. Such structure is present in supply chain product flows problems where decisions typically involve determining, for each period of time, the volume of products that needs to be transported between business units, the target inventories to keep on hand, the amount to replenishment from suppliers, the quantities of each product to ship to customers and the production quantities [12, 13].

These problems are typically solved using commercially available software like IBM ILOG CPLEX. They can solve problems having several hundreds of thousands variables in a few minutes, thanks to well known algorithms like Dantzig's *simplex* method.

One shortcoming of these solvers is that they usually return one and only one solution for a given problem (defined as a set of variables, constraints and an objective function) – although thousand of alternative optimal solutions may exist. Most of the time, the returned solution is inadequate as the problem specification does not take into account the contextual and political information known only to the decision maker [4]. It is indeed very difficult in a mathematical model to consider all the preferences of the user. Moreover, the decision maker does not know exactly all the constraints; he "discovers" some when he sees solutions violating them.

Although the solution produced by the solver often contains hundreds of thousands of variables, in practice the decision-maker analyzes the solution by consulting only a few charts, each one containing a few dozens of variables (e.g. 52 variables matching the 52 weeks in the year). An example of this type of chart is provided in Figure 1.

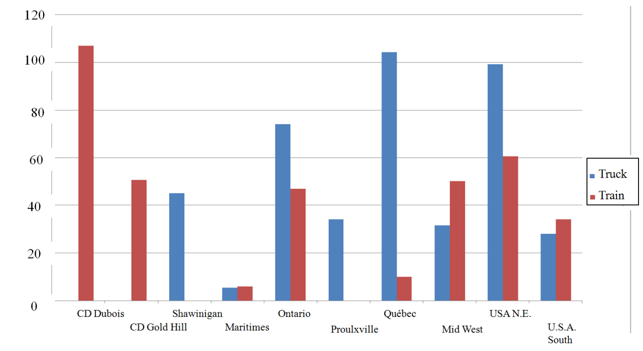
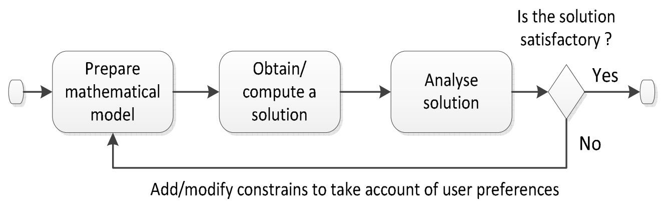


Figure 1. Example of a chart used by decision-makers to analyze the solution of a supply chain optimization problem.

When the solution does not satisfy the requirements of the decision maker (e.g. he does not like the value for a specific variable), the mathematical model has to be modified and reoptimized so that a new solution can be found (Figure 2).



Is the solution satisfactory ?
Prepare mathematical model
Obtain/ compute a solution
Analyse solution
Yes
No
Add/modify constrains to take account of user preferences

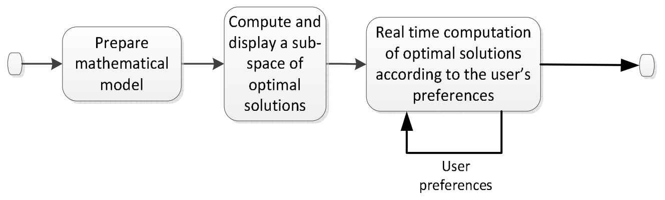
Figure 2. Typical process of finding a suitable solution

Not only is this process very long (solving an integrated supply chain planning problem may easily take up between 10 and 60 minutes), this is also a frustrating iterative process because when the decision-maker is not satisfied by a variable value, he never knows if the variable is really constrained to that specific value in order for the solution to be optimal, or if exists other optimal solutions with other values for that variable. He does not have the choice but to run new optimizations.

III. The Proposed Mixed-Initiative System for Linear Optimization Problems

A good Mixed-Initiative System for linear optimization would provides the user with information regarding whether or not the variables of interest are constrained to a certain value in order for the solution to be optimal. It also allows the user to interactively modify the value of a variable and see, in real-time, how the other variables should be modified in response to that modification.

The system we propose 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 sub-space of optimal solutions (Figure 3).



Prepare mathematical model
Compute and display a sub-space of optimal solutions
Real time computation of optimal solutions according to the user's preferences
User preferences

Figure 3. MIS concept applied to linear programming

A. *Obtaining and displaying a sub-space of optimal solutions*

To palliate the fact that classical solvers return a single solution, we proceed as follows. We use a classical solver to compute an optimal solution for the problem *P*. Once the optimal objective value is known, we search for other optimal solutions. We create a new instance *P'* by augmenting the instance *P* with 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 on the graph, we search 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 $2n$ computations can be performed in parallel using a supercomputer.

Then, one can display on the chart (see Figure 4) the *range of optimality* for each variable x_i (the minimum and maximum values the variable can take in an optimal solution).

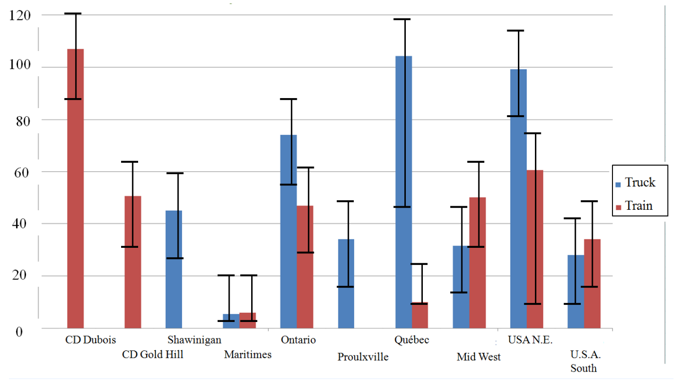


Figure 4. An optimal solution with range of optimality for the variables.

Moreover, it is possible to construct an "average" solution to be displayed to the user by summing the $2n$ solution vectors and dividing the result by the scalar $2n$. The validity of this process relies on the concept of convexity that we present bellow.

A n-dimensional vector represents the values assigned to n variables, and therefore a solution to a problem. A vector also represents a point in an n-dimensional space. Let x and y be two n-dimensional vectors. A linear combination is the weighted sum of two or more vectors, e.g. $\alpha x+\beta y$. A linear combination is affine if the sum of the weights equals one, e.g. $\alpha x+(1-\alpha)y$. Any point on the line passing by the points x and y is an affine combination of x and y. A linear combination is convex if it is affine and if each weight is non-negative, e.g. $\alpha x+(1-\alpha)y$ for $\alpha\in[0,1]$. The points that can be obtained by a convex combination of x and y are those on the segment connecting x to y. A space S is convex if any convex combination of two points in S is also in S.

Theorem 1 is a well-known result in the linear programming literature (e.g. [14]). It shows how an optimal solution to a linear program can be obtained from the convex combination of distinct optimal solutions. We reproduce the proof of the theorem as it shows the properties we will exploit in our system.

Theorem 1 : The convex combination of two optimal solutions of a linear program is also an optimal solution.

The space $S=\{x \mid Ax\le b,\ x\ge 0\}$ contains the feasible solutions of a linear program. Let $x\in S$ and $y\in S$ be two solutions. We show that any convex combination of x and y belongs to S.

$$\begin{aligned} A(\alpha x+(1-\alpha)y) &= \alpha Ax+(1-\alpha)Ay \\ &\le \alpha b+(1-\alpha)b \qquad \text{since } \alpha\in[0,1] \\ &= b \end{aligned}$$

Suppose that x and y are two optimal solutions to the linear program, i.e. x maximizes the scalar product $c^{\mathrm{T}}x$ and so does y. Let $c^*=c^{\mathrm{T}}x=c^{\mathrm{T}}y$ be the optimal cost. We show that any convex combination of x and y is also an optimal solution. □

The convex hull of a set of points $X=\{\vec{x}_1,\dots,\vec{x}_n\}$ is the space containing all convex combinations of the points in X. Formally, we note

$$\operatorname{conv}(\vec{x}_1,\dots,\vec{x}_n)=\left\{\sum_{i=1}^{n}\alpha_i\vec{x}_i \,\middle|\, \alpha_i\ge 0,\ \sum_{i=1}^{n}\alpha_i=1\right\}$$

From Theorem 1, if the points $\vec{x}_1,\dots,\vec{x}_n$ are optimal feasible solutions to a linear program, then all points in $\operatorname{conv}(\vec{x}_1,\dots,\vec{x}_n)$ are optimal feasible solutions.

In our system, the convex hull of the solutions $\underline{s_i}$ and $\overline{s_i}$ that minimize and maximize each of n variables form a subspace of optimal solutions.

B. *Real-time convex combination of optimal solutions*

A chart like the one depicted in Figure 4 allows the user to see the flexibility (range of optimality) he has for any of the variable 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. The system should then adjust the value taken by the other variables such that the solution is still optimal. Normally, this operation would require a lengthy complete re-optimization of the problem. Based on Theorem 1, we know however that it is possible to compute a new optimal solution by generating new 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.

If this could be done in real-time, we could instantly refresh the chart and display the impact of the user modification on the other variables. The user could thus *navigate* through the solutions space, successively modifying several variables until a suitable solution is found.

Figure 5 illustrate this idea for a small problem with two variables. The polygon shows the set of convex combination formed by optimal solutions minimizing and optimizing each of the variables[1].

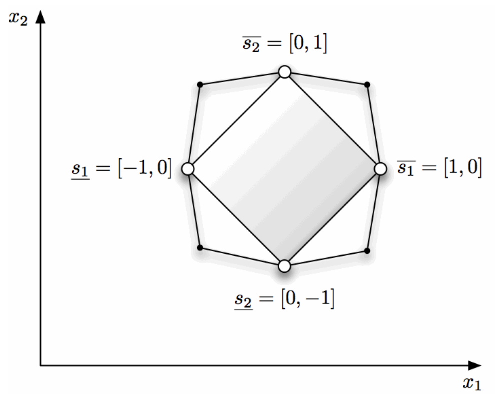


Figure 5. Available optimal zone using convex combinations.

When the user modifies a variable, we do not just want to find any point in the sub-space such that the modified variable takes the desired value; we want to find a solution such that the other variables move as little as possible so that the overall system appears to be *stable*. If it would not be the case, then any modification made to a variable could offset the previous modifications made to the other variables and it would never be possible for the user to converge to a satisfactory solution.

Given a solution x, a variable x_i, and a value v, finding a new solution x' where $x'_i=v\ne x_i$ while changing as little as possible the values of the other variables is an optimization problem.

[1] This example (Figure 5) also shows that there could be optimal solutions not covered by this sub-space. This is why we say that the user navigate in a space (formed by convex combination of our extreme solutions) that is in reality a sub-space of the optimal solutions.

We propose four approaches to solve this optimization problem. Each of them is a trade-off between *responsiveness* and *stability*.

Definition 1: *Responsiveness* refers to the computation time needed to obtain a new solution each time the decision-maker modifies a variable. It determines the time before the software can display the new optimized solution.

Definition 2: *Stability* is the property of computing a new solution that is as close as possible from the current solution.

While responsiveness guarantees that the software reacts in real-time to the changes of the decider, stability ensures that the software does not move the solution away from the choices that the decider previously made.

Responsiveness is measured in milliseconds. Stability is given by the Euclidean distance between the original solution x and the modified solution x'.

Since responsiveness is an important criterion, we exclude the method of resolving the original problem from scratch. We restrict ourselves to find a solution in the convex hull of the $2n$ precomputed solutions $\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n}$, a much smaller problem then the original one. Here are the proposed methods.

1. Minimizing the Euclidean distance. This method aims at finding in the convex hull $\operatorname{conv}(\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n})$ the solution whose Euclidean distance is the closest to x. Let S be the $n \times 2n$ matrix whose columns are the solutions $\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n}$. The solution we are looking for is a convex combination of these solutions. We are therefore looking for a vector α such that $x' = S\alpha$, $\sum_{i=1}^{2n} \alpha_i = 1$, and $\alpha_i \geq 0$. Moreover, we want x'_i to be equal to v. We obtain the following quadratic problem where e_i is the vector with null components except for the ith one that is equal to one, $\vec{1}$ is the vector with all components equal to one, and $\vec{0}$ is the null vector:

$$\begin{array}{lrcl} \min & \|x - S\alpha\|^2 & & \\ \text{s.t.} & e_i^T S\alpha & = & v \\ & \vec{1}^T\alpha & = & 1 \\ & \alpha & \geq & \vec{0} \end{array}$$

The squared Euclidean distance $\|x - S\alpha\|^2$ can be rewritten as $\alpha^T S^T S\alpha - 2x^T S\alpha + x^T x$. The problem is a semidefinite program. The objective function is quadratic and convex and the constraints are linear. Solvers like CPLEX can solve this problem using interior point algorithm.

2. Minimizing the maximum distance. Even though semidefinite problems can be efficiently solved, they are more computationally demanding than linear programs. We propose another approach that is likely to be faster. Rather than minimizing the Euclidean distance between the new and the former solution, we minimize the maximum distance between two variables, i.e. we minimize $g = \max_j |x_j - x'_j|$. The linear program we present is inspired from the previous one. Once more, we compute the vector α that expresses x' in terms of a convex combination of the $2n$ solutions stored in the columns of S ($x' = S\alpha$). The two first constraints of this linear program force the variable g to be at least as large as $|x'_j - x_j|$. The unknown of this problem are the vector α and the gap variable g.

$$\begin{array}{lr} \min & g \\ \text{s.t.} & S\alpha - \vec{1}g \leq x \\ & -S\alpha - \vec{1}g \leq -x \\ & e_i^T S\alpha = v \\ & \vec{1}^T\alpha = 1 \\ & \alpha \geq \vec{0} \end{array}$$

This problem can be solved using the simplex method, which is usually faster than the interior point method used for semidefinite problems of the previous approach. The obvious drawback is that the objective function is not completely in line with our stability metric. Even if we minimize the distance between the two variables that are the furthest apart, we do not minimize the distance between the other variables. A solution x' where all variables are changed by the same amount would be equivalent to a solution where only the variable x_i is modified.

3. A bipolar heuristic. We introduce a third method that focuses on responsiveness. The *bipolar heuristic* is a very fast way to compute a new solution. Given that the decider wants to set the variable x_i to the value v, we compute the unique convex combination $x' = \alpha\underline{s_i} + (1 - \alpha)\overline{s_i}$ such that $x'_i = v$. To do so, we set $\alpha = (\overline{s_{ii}} - v)/(\overline{s_{ii}} - \underline{s_{ii}})$ where $\underline{s_{ii}}$ and $\overline{s_{ii}}$ are the values of x_i in the precomputed solutions that minimize and maximize x_i. The geometric interpretation of this method is that we choose x' to be the intersection of the segment connecting $\underline{s_i}$ to $\overline{s_i}$ and the plane $x'_i = v$.

This solution is by far the most responsive one, as it does not involve any solver. However, it does not take into account the stability metric. Indeed, a small change on the variable x_i can produce big changes on the other variables of the solution. For example, consider an example in the plane where we have $\underline{s_1} = [-1, 0]$, $\overline{s_1} = [1, 0]$, $\underline{s_2} = [0, -1]$, and $\overline{s_2} = [0, 1]$ (see Figure 6). Let the current solution be $x = [0.5,\ 0]$. A small move of variable x_2 from 0 to ε causes the solution to change for $x' = [0,\ \varepsilon]$ even though the solution $x' = [0.5,\ \varepsilon]$ is a better choice.

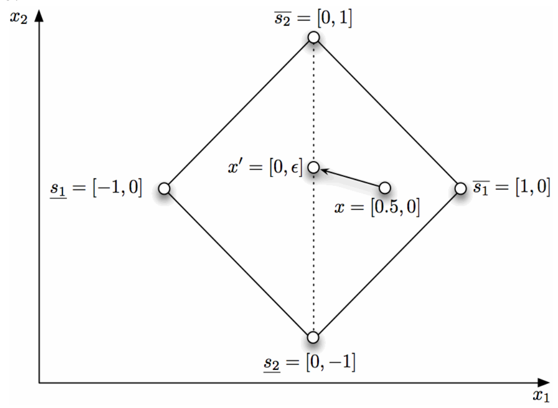

x_2
$\overline{s_2} = [0, 1]$
$x' = [0, \epsilon]$
$\underline{s_1} = [-1, 0]$
$\overline{s_1} = [1, 0]$
$x = [0.5, 0]$
$\underline{s_2} = [0, -1]$
x_1

Figure 6. Illustrating the bipolar method stability drawback.

4. The triangular heuristic. To palliate to the non-stability of the previous approach, we propose the *triangular heuristic*. First, this method determines whether the variable x_i is increased ($v > x_i$) or decreased ($v < x_i$). If the variable is increased, we compute the unique convex combination between the current solution x and $\overline{s_i}$ that intersects the plane $x_i = v$. If the variable is decreased, we compute the unique convex combination between the current solution x and $\underline{s_i}$ that intersects the plane $x_i = v$. In other words, if the decision-maker increases x_i, we set

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

where

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

If the decider decreases the variable x_i, we set

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

where

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

Going back to the example of Figure 6 where $x = [0.5, 0]$. Increasing x_2 by ε sets the new solution to $x' = [0.5 - \varepsilon/2,\ \varepsilon]$. This is significantly more stable that the bipolar heuristic.

There is a second difference between the bipolar heuristic and the triangular heuristic. The bipolar heuristic can only produce a solution that lies on a segment connecting a solution $\overline{s_j}$ to a solution $\underline{s_j}$ for a given j. The triangular heuristic can produce any solution in the convex hull $\text{conv}(\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n})$. Indeed, there exist a sequence of movements that the decider can make in order to reach any solution in the convex space.

Property 1: A sequence of movements made with the triangular heuristic from a solution x can reach any solution in the convex hull $\text{conv}(\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n})$.

Proof. Let x' be the final solution in $\text{conv}(\underline{s_1}, \overline{s_1}, \ldots, \underline{s_n}, \overline{s_n})$. Let S be the matrix whose columns are the extreme points of the convex hull. There exists a non-negative vector α whose components sum to one and that satisfies $x' = S\alpha$. We prove by induction on the number of non-null components in α that any x' is reachable from a sequence of triangular movements.

Suppose that there exists a single non-null component in α. This component is equal to one and we have $x' = \overline{s_j}$ or $x' = \underline{s_j}$ for some j. Increasing the value of x_j in the current solution to its maximum or minimum makes the triangular heuristic fix the new solution to x'.

Suppose that one can reach any solution $x = S\alpha$ with $k > 0$ non-null components in α, we prove that one can also reach any solution $x' = S\beta$ with $k + 1$ non-null components in β. Let $x' = S\beta$ be a solution with $k + 1$ non-null components in β. Let j be the index of any non-null component in β. We construct the vector α as follows.

$$a_i = \begin{cases} 0 & \text{if } i = j \\ \frac{\beta_i}{1-\beta_j} & \text{if } i \neq j \end{cases}$$

The solution $x = S\alpha$ is reachable since it has only k non-null components. Moreover, let $s \in \{\underline{s_k}, \overline{s_k}\}$ be the solution associated to the component β_j. It is possible to move from the solution x to x' using the triangular heuristic since this relation holds.

$$x' = (1-\beta_j)x + \beta_j s = S\beta$$

This is the relation computed when the decider changes the value of x_k to x'_k. □

Property 1 ensures that any solution in the convex hull can be reached.

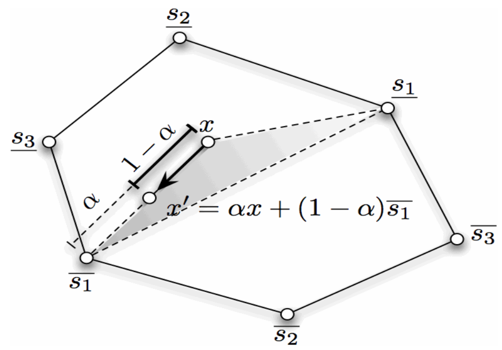


Figure 7. An example of the triangular method.

IV. EVALUATION

A. The industrial case study

The industrial partner in this project is looking to put in place an efficient process for integrated sales and operation planning (S&OP). As part of this process is the use of a large scale linear programming model with over 200,000 variables and 100,000 constraints we introduced in [13]. This model generates a unified tactical plan for the production and distribution network. This plan is divided in fifty two weekly periods and integrates decisions pertaining to production, distribution and sales.

Production. Lumber production from sawmills involves three main processes: sawing, drying and planning. The sawing process is divergent as a single product entering the production line gets transformed into several different products. Different production recipe can be chosen in order to influence the products' basket that gets produced. The drying of the wood allows giving its stability over its life time. The drying is done by batch in large kilns and the process can lasts from 2 to 5 days depending on several factors like the species, the lumber dimension (section width and thickness) and the original moisture content. Kiln drying may be preceded by an air drying operation that can lasts from 2 to 18 weeks. Air drying usage reduces kiln dry time and may increase the lumber quality. The reduction in kiln dry time depends not only on the time the wood was air dried, but also the period of the year at which the air drying was done. Finally, the planing is the process that gives the lumber its finish and exact dimension. Because of the wood characteristics and defects, not two lumber units entering the planing process are exactly identical. It is therefore impossible to precisely predict the final products exiting the planing process given a set of input products. Historical production planing data are used to estimate the production of a given input product. The tactical production planning help determine, for each sawmill in the network and for each week, the sawing recipe to use, the volume of each sawed product to air dry along with the air dry duration, the volume of each product to kiln dry and the volume of each product to plane along with the planing recipe to use.

Sales. In the North American forest products industry, lumber market prices follow a seasonal pattern. The planning model takes this into account and may suggest producing in advance some quantities of a given product that will be sold at a later time during the year when the market price is more profitable. At the planing stage, the model will suggest the most appropriate recipe considering the selling prices for that period. The model not only considers the varying selling price in time. It also considers the limited volume that can be sold in given markets per product per period. The user running the model can also constrains the percentage of the production that goes to service each market. Another complexity comes from the fact that the drying operation is optional and that some customers may want to buy green planed lumber at a discounted price. So when the drying capacity is reached but there is still planing capacity, lumber can be routed directly from the completion of the sawing operation to the planing activity.

Distribution. The company runs three sawmills in the province of Quebec in Canada and makes use of two distribution centers in the eastern United States and one in Quebec. It is not always required for the lumber products to go through a distribution centre to reach a market, but it is sometimes more economical. Transportation can also occur between mills as in the company' network, not all sawmills have planing capability. From a sawmill with no planing capability, the dry lumber can either be moved to another sawmill with planing capability or it can be transported to a distribution centre or be sent directly to a customer. The model help determine where to send the lumber along with the transportation mode to use which can be either by trucks or by rail.

Planning the overall network of sawmills taking into consideration market requirement, price fluctuation, production capacity and transportation alternatives and costs is not an easy task. To solve to optimality this complete problem can take up to 45 minutes using CPLEX, one of the leading software to solve linear programming models.

B. *Experiments*

The main purpose of these experimentations is to evaluate the behavior of the system in terms of stability and responsiveness metrics (as defined in Section III.B) for the proposed approaches.

We first generated an optimal solution for the industrial case introduced previously. Then, we simulated the situation where the decision-maker visualizes various charts with different number of variables on them (1, 10, 20, 30, 40, and 52 variables). For each variable we computed its optimality range. We then simulated the situation where the decision-maker asks for modification of some variable values. We tested the behavior of the system for small modifications (the variable is increased or decreased by a gap corresponding to 7% of its optimality range, i.e. the difference between the maximum and minimum a variable can take), medium modifications (45% of optimality range) and large modifications (75% of optimality range).

After each modification, an optimal solution is computed using one of the four approaches and we measure responsiveness (computation time) and stability (Euclidean distance between original and recomputed solutions).

C. *Results*

Figure 8 provides results for responsiveness of the system (in milliseconds). We recall that solving the original problems each time a modification is asked by the user would require a few minutes for each modification. Using the proposed approaches the worst case observed was a recomputation time of 600 milliseconds (minimization of the Euclidean Distance approach), see subfigure i. We were not expecting this method to perform so well because of the computation time required to solve a quadratic optimization problem. Nonetheless, it appears the problem is small enough to provide good performance.

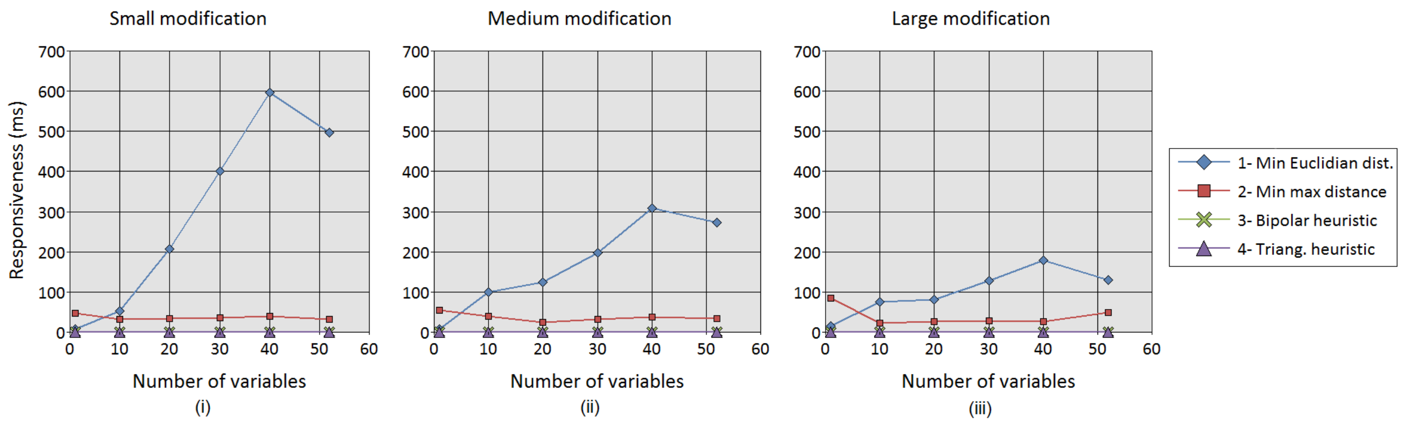


Figure 8. Solution recomputation time (in milliseconds) in reaction to small (i), medium (ii) and large (iii) variable value modifications, according to the number of variables displayed.

However, as expected, computation time rapidly grows with the number of variable displayed on the charts. Clearly, in a situation where the decision maker would be visualizing 4 or 5 charts with 52 variables on each of them, the Euclidean approach would not meet our real-time expectations.

Subfigures i, ii, and iii show that the size of the modification asked by the user has a strong effect on the responsiveness of the Euclidean approach. The larger the modification is (in terms of % of the optimality range) the easier is the computation of the new solution. The explanation is the following. In the extreme case where the user constrains the variable to take its value $\overline{x}_i$, the system is limited to the solutions corresponding to an extreme point of our convex hull. Said otherwise, there is not much exploration to perform.

The responsiveness of the method that minimizes the maximum distance is not affected by how much a variable is changed. Its linear optimization problem is small and easy to solve – although it is of not as efficient as the heuristic methods. However, those three approaches meat our real-time expectations.

Figure 9 shows the Euclidean distance between the original and the recomputed solution after a small (i), medium (ii) or large (iii) modification and according to the number of variables displayed. The method that minimizes the Euclidean distance always offer the best results.

As expected from Section III.B, the bipolar heuristic gives the worst results. Small modifications of a variable value (subfigure i) lead to huge modifications to other variables. Even worse, the beta-testers considered unacceptable that the new solutions are computed without any consideration for changes they made previously.

For small modifications (subfigure i), minimization of the maximum distance or using the triangular heuristic provides stability very close to the Euclidean approach. Actually, for the minimization of the maximum distance, results on the chart are indistinct from those of the Euclidean approach.

However, minimizing the maximum distance sometimes leads to the problematic situation discussed in Section III.B, i.e. the largest change is minimized but the other variables move for no valid reason. The triangular heuristic does not have this drawback.

We analyze the impact of small, medium and large modifications (comparing subfigures i, ii and iii). The larger the modification asked by the user is, the more the different approaches give similar results. As discussed previously, large modifications to a variable value lead to a very small space of optimal solutions to which all the approaches are restricted.

V. Conclusion

Supply chain optimization leads to large problems with hundreds of thousands variables. They however can be solved in reasonable time (less than an hour) as they often show a linear structure. However, this computation time is unacceptable for iterative and interactive decision-making contexts where decision-makers need to provide their input.

In this research, we proposed a system that provides the decision-makers with a representation of an optimal solution space directly on the charts they are used to work with. This space is modeled as a convex hull of optimal solutions minimizing/maximizing the variables of interest. Users can interactively modify the value of a variable and the system is able to recompute a new optimal solution in less than a second (instead of optimizing the original problem, which is very long).

We developed four approaches for this real-time re-optimization. The minimization of the maximal distance and the bipolar heuristics both have drawbacks in terms of solution stability. The minimization of the Euclidean distance is optimal regarding this aspect. It should be used in situations where the decision-makers based their decision on the analysis of a small subset of variables (typically a chart with a dozen of variables) although the optimization problem can have hundreds of thousands of variables. If the decision-makers need to simultaneously analyze and modify charts with hundreds of variables, the triangular heuristic should be used as our experiments reported near optimal performance.

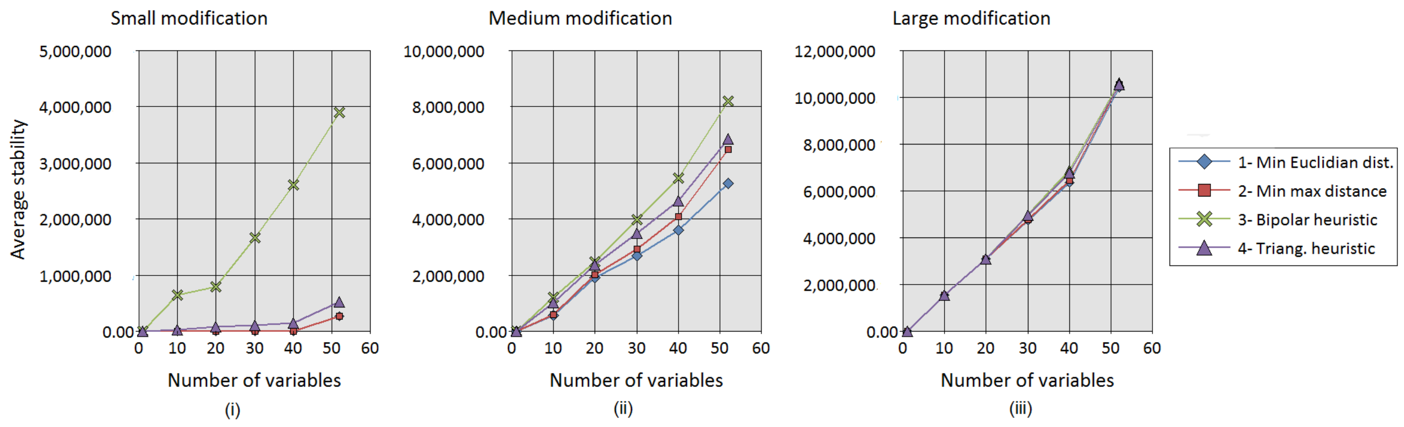


Figure 9. Euclidean distance between the original and recomputed solution after a small (i), medium (ii) or large (iii) variable value modification, according to the number of variables displayed.

References

[1] M. A. Hearst, "Mixed-Initiative Interaction", *IEEE Intelligent Systems*, vol. 14, pp. 14, 1999.

[2] J. E. Allen, "Mixed-initiative interaction", *IEEE Intelligent Systems*, vol. 14, pp. 14-16, 1999.

[3] M. Fleming, "The use of increasingly specific user models in the design of mixed-initiative systems", Proceedings of the 17th Conference of the Canadian Society for Computational Studies of Intelligence (LNCS #3060), London, Canada, 2004, pp. 434-438.

[4] G. Klau, N. Lesh, J. Marks, and M. Mitzenmacher, "Human-guided search", *Journal of Heuristics*, vol. 16, pp. 289-310, 2010.

[5] W. Kun and W. S. Havens, "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), Victoria, Canada, 2005, pp. 79-90.

[6] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J. C. J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, R. Kanna, 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*, vol. 19, pp. 8-12, 2004.

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

[8] J. L. Bresina and P. H. Morris, "Mixed-initiative planning in space mission operations", *AI Magazine*, vol. 28, pp. 75-88, 2007.

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

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

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

[12] A. G. de Kok and S. C. Graves, *Supply Chain Management: Design, Coordination and Operation*. Amsterdam: Elsevier, 2003.

[13] P. Marier, J. Gaudreault, M. Knaptik, and M. Savard, "Maximizing profits through sawmills specialization and supply chain design", Proceedings of the 14th Annual International Conference on Industrial Engineering Theory, Applications & Practice, Anaheim, CA., 2009, pp. 484-491.

[14] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, N.Y.: Dover, 1998.