10^{th} *International Conference of Modeling and Simulation - MOSIM14 – November 5-7 - Nancy - France*
"Toward circular Economy"

RE-PLANNING WITH MINIMAL PERTURBATION

Thierry Moisan, Claude-Guy Quimper, Jonathan Gaudreault & Sébastien Michaud

FORAC Research Consortium, Université Laval
2325, rue de l'Université
Québec (Québec) G1V 0A6 - Canada
Thierry.Moisan.1@ulaval.ca, Claude-Guy.Quimper@ift.ulaval.ca,
Jonathan.Gaudreault@ift.ulaval.ca, Sebastien.Michaud.2@ulaval.ca.

ABSTRACT: *Mixed Integer Programming (MIP) models are known to be sensitive to slight changes. In this study, we measure the perturbation to a production plan that occurs when re-planning is carried-on due to new orders arrival. Perturbation level is measured using well known metrics: (1) the Hamming distance, (2) the edit distance, and (3) the Damerau-Levenshtein distance. We propose three alternative formulations of the MIP model that allows for the integration of these metrics directly into the objective function. By simulating arrival of new orders for a lumber finishing mill, we show how these new re-planning models reduce the perturbation and outperforms the approach of using the original planning model.*

KEYWORDS: *Re-planning, Plan repair, Persistence, Hamming distance, Edit distance, Damerau-Levenshtein distance.*

1 INTRODUCTION

Most research in planning and scheduling proposes models and algorithms that compute the production plans "from scratch" : the aim of the model is simply to come up with the plan that best satisfies the needs of the customers, with no consideration for the previous computed plans. However, planning algorithms/models are generally used in industrial practice following a *rolling horizon* approach (Linus 1991). As an example, the mill can plan in advance for the next 15 days, but needs to re-plan every day (or even each time a new order arrives). Each time re-planning is carried-on, one tries to satisfy new orders at best while being constrained by the promises from previous orders. This has some drawbacks : it leads to frequent modifications/perturbations of the production plan that needs to be communicated to the staff on the floor (Brown, Dell & Wood 1997).

Less perturbation is preferable as it allows to prepare the work in advance more easily. Among many advantages, it allows the material used in the operations to be moved beforehand to the production facility.

In this paper, we show how a planning/scheduling MIP model can be modified to minimize perturbation in a re-planning context. The *old* plan is passed as an input parameter, and the new objective function minimizes the distance between the *old* and the *new* production plan.

Section 2 presents preliminary notions regarding metrics of stability/distance in the context of production planning and order promising. The simplest metric is called the *Hamming distance* and has already been encoded into a MIP (Brown, Cormican, Lawphongpanich & Widdis 1997). However, to the best of our knowledge, the more refined distance metrics (edit distance and Damerau-Levenshtein) have never been encoded into a MIP. Therefore, we propose a formulation in Section 3 and 4. Finally, in Section 5, we perform experiments for a lumber finishing planning/scheduling problem. By simulating the arrival of new orders for a lumber finishing mill, we show how these new re-planning models reduce the perturbation and outperforms the approach of using the original planning model.

2 PRELIMINARY CONCEPTS

We review related planning and scheduling approaches then describe multiple distance measures and their use.

Planning is the operation of deciding which activities should be executed to reach a goal. Scheduling is the operation of deciding when activities are executed and which resource executes an activity. In a planning and scheduling problem, both planning and scheduling are simultaneously achieved. Doing both operations at the same time has the advantage to choose the activities while taking into account the availability of the

resources. It results in a better solution at the cost of solving a far more complex problem. In production planning and scheduling problems, an activity is a process that consumes input products and transforms them into output products. An inventory of all input and output products is maintained. The quantity of a product in the inventory must be non-negative. Orders are requirements for a product to appear in the inventory in sufficient quantity. If the quantity is not met, we are facing order lateness. A production plan is an ordered sequence of jobs that usually optimize an objective function such as the minimization of order lateness.

The production plan needs to be regularly updated as new orders are received. Before promising a new order, a decider needs to know what is the impact of this new order on the production plan. The impact can either be a change of objective value or a too big reorganization of the work. It is desirable to compute a new production plan that is similar to the original one.

2.1 RE-PLANNING

Two main approaches exist to adapt a production plan to changes: re-optimizing from scratch or repairing the existing plan. Fox et al. (Fox, Gerevini, Long & Serina 2006) compared these two approaches and concluded plan repair can offer the best plan stability. Plan repair typically uses local search: the local search starts from the original plan and improves it by exploring the solution space near the original plan.

Persistence is the concept of using information of a previous optimization to accelerate and stabilize a new optimization. We suppose that both optimizations share common partial results. The partial results of the first optimization are added to the model to facilitate the second optimization. Persistence also has the property to keep the solution set stabler: a small change is less likely to create a completely different solution. Brown et al. (Brown, Dell & Wood 1997) offer a good introduction to optimization with a persistence incentive.

Juin-han Chen and Chin-tai Chen (Chen & Chen 2009) used a multi-phase approach to promise orders in two phases. The first phase reserves the needed resources and the second phase promises the orders based on the reserved resources. The reservation mechanism that the authors propose also helps to stabilize the plan. However, in some cases, finding the resources to reserve can be a difficult problem by itself, for example when co-production occurs.

In planning and scheduling problems, it is likely that one wants to modify an existing plan to fulfill new orders that are coming in. This new production plan should not delay the orders that were already promised. A persistence incentive can be added by reusing the information about the inventory from the previous production plan to ensure that the original orders are still guaranteed when re-optimizing.

Let $I_{r,t}$ be the quantity of product r at period t that is in the inventory in the original production plan and $I^+_{r,t}$ be the quantity in the new production plan for the same product and period. Let $O_{r,t}$ be the quantity newly ordered for product r required for period t. The difference between the new and old inventory of product r must be at least the quantity of the new orders of this product.

$$I^+_{r,t} \geq I_{r,t} - \sum_{j=1}^{t} O_{r,j} \qquad \forall r, t \qquad (1)$$

Constraint 1 guarantees that accepting new orders $O_{r,j}$ attributes no additional delays to existing orders by forcing the new inventory to contain at least as much finished products as the original inventory. Moreover, we can choose, as an objective function, to minimize the backorders of the new orders. We can also minimize the differences between the previous production plan and the new one being computed according to various distance metrics while keeping the current production. We present in the next subsections different distance measures.

2.2 Perturbation / Distance measures

A sequence a is an ordered list of (possibly redundant) elements that we call characters. The characters are taken from a set of characters Σ called the *alphabet*. For instance, a production plan can be represented with the sequence of jobs ordered in chronological order of execution. We denote by a_i the i^{th} character of the sequence a and by $|a|$ its length. A distance $d(x, y)$ is a function that accepts two sequences and returns a number that characterizes how different the sequences are. The distance is null if and only if the two sequences are identical. It is positive if the sequences are different. The greater the distance is, the more differences there are between the two sequences. The distances studied in this paper are metrics and therefore they satisfy the triangle inequality, i.e. for any three sequences x, y, and z, the relation $d(x, y) + d(y, z) \geq d(x, z)$ holds.

Distance measures between two sequences are typically used in the context of spell checking to find which word in the dictionary is the closest to the one that is typed (Kukich 1992). Sankoff and Kruskal (Sankoff & Kruskal 1983) give many other applications examples of this type of metrics, including coding theory, human speech recognition, and com-

putational biology. In our case, we consider the production plans as sequences and we measure their distance with three metrics that we present in this section: the Hamming distance, the edit distance, and the Damerau-Levenshtein distance.

2.2.1 Hamming distance

The Hamming distance (Hamming 1950) is defined on two sequences that have the same length and computes the number of elements that are pair-wise different. Let $I(p)$ be the function that returns 1 if the proposition p is true and 0 otherwise. The hamming distance over two sequences a and b is defined as follows.

$$H(a,b) = \sum_{i=1}^{n} I(a_i \neq b_i). \tag{2}$$

2.2.2 Hamming distance minimization

Brown et al. (Brown, Cormican, Lawphongpanich & Widdis 1997) use the Hamming distance to minimize the turbulence of a submarine berthing plan when it is revised to include new needs.

Their approach supposes that a plan already exists. New orders are coming in and must be taken into account with new optimization. Additional constraints are added to the original optimization model to compute the Hamming distance. The original plan is represented by a fixed sequence a and new decision variables are added to represent the new plan b. The new objective function minimizes the Hamming distance between sequence a and sequence b. To add a persistence incentive, constraints such as the one explained in Section 2.1 can be included.

The authors model the problem as a linear program. They introduces a binary variable $q_i = I(a_i \neq b_i)$ that indicates whether the characters a_i and b_i are different and minimizes the sum of these variables. The following model minimizes the Hamming distance between a and b.

$$\min \sum_{i=1}^{n} q_i \tag{3}$$

$$a_i - b_i \leq |\Sigma| q_i \qquad \forall\, i \in 1..n \tag{4}$$

$$b_i - a_i \leq |\Sigma| q_i \qquad \forall\, i \in 1..n \tag{5}$$

$$q_i \in \{0,1\} \tag{6}$$

Constraint (4) and (5) ensure that $q_i = 0$ implies that $a_i = b_i$. Note that one could obtain $q_i = 1$ and $a_i = b_i$ in a feasible solution, but such a solution is not optimal and is therefore discarded by the solver.

2.2.3 Edit distance

A finer distance measure is the edit distance (Levenshtein 1966), also known as the Levenshtein distance. While the Hamming distance counts the number of substitution that needs to be applied on a sequence a to obtain a sequence b, the edit distance also counts the number of insertions and deletions. For example, the sequences "abcdef" and "bcdefa" have a Hamming distance of 6. However they have an edit distance of 2: the character "a" is deleted at the beginning of the string and inserted at the end. The edit distance allows to compare sequences of different sizes.

Let $d_{i,j}$ be the edit distance between the subsequence formed with the i first characters of a sequence a and the subsequence formed with the j first characters of a sequence b. The edit distance between the two sequences a and b is given by $d_{|a|,|b|}$ and can be recursively computed as follows.

$$d_{i,j} = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{l} d_{i-1,j} + 1, \\ d_{i,j-1} + 1, \\ d_{i-1,j-1} + I(x_i \neq y_j) \end{array} \right\} & \text{otherwise} \end{cases} \tag{7}$$

Computing the edit distance directly from this recursion takes exponential time. However, a dynamic programming algorithm can compute the matrix d by computing its values row by row in $O(|a||b|)$ time.

Figure 1 depicts the matrix created by the dynamic programming approach. a_0 and b_0 represent empty sequences and $d_{i,j}$ represents the edit distance between $a_1 \dots a_i$ and $b_1 \dots b_j$. Hence, to obtain the complete edit distance between these two sequences, one needs to compute $d_{|a|,|b|}$. This matrix allows to use recursion 7 without having to compute the values $d_{i,j}$ multiple times.

		a	b	c	d	e	f
	0	**1**	2	3	4	5	6
b	1	1	**1**	2	3	4	5
c	2	2	2	**1**	2	3	4
d	3	3	3	2	**1**	2	3
e	4	4	4	3	2	**1**	2
f	5	5	5	4	3	2	**1**
a	6	5	6	5	4	3	**2**

Figure 1: The matrix used to compute the edit distance between "abcdef" and "bcdefa".

Using a similar approach, Wagner (Sankoff & Kruskal 1983) presents an automaton with costs on transitions that computes the edit distance.

2.2.4 Damerau-Levenshtein distance

The Damerau-Levenshtein distance (Damerau 1964) further improves the edit distance by allowing one more edition: the transposition of two consecutive characters. For example the strings "abc" and "bac" have a Hamming and an edit distance of 2 but have a Damerau-Levenshtein distance of 1. Let $d_{i,j}$ be the Damerau-Levenshtein distance between the subsequence formed with the i first characters of a sequence a and the subsequence formed with the j first characters of a sequence b. As for the edit distance, the Damerau-Levenshtein distance can be computed recursively as follows.

$$d_{i,j} = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{l} d_{i-1,j} + 1, \\ d_{i,j-1} + 1, \\ d_{i-1,j-1} + I(x_i \neq y_j), \\ d_{i-2,j-2} + 1 \end{array} \right\} & \text{if } i,j > 1 \wedge \\ & a_i = b_{j-1} \wedge \\ & a_{i-1} = b_j \\ \min \left\{ \begin{array}{l} d_{i-1,j} + 1, \\ d_{i,j-1} + 1, \\ d_{i-1,j-1} + I(x_i \neq y_j) \end{array} \right\} & \text{otherwise} \end{cases} \tag{8}$$

As for the edit distance, computing the Damerau-Levenshtein can be done in time $O(|a||b|)$ using dynamic programming by filling in the matrix d row by row.

To the best of our knowledge, the edit distance and the Damerau-Levenshtein distance have never been encoded into a MIP. We present in Section 3 a model that minimizes the edit distance and in Section 4 a model that minimizes the Damerau-Levenshtein distance.

3 A MIP MODEL THAT MINIMIZES THE EDIT DISTANCE

Consider two sequences of variables a and b. These sequences are unknown and are subject to different constraints. We want to minimize the edit distance between a and b. We model this objective as a mixed integer program.

We reduce the problem of computing the edit distance to the problem of finding a shortest path in a directed acyclic graph. We define a graph for which there is a node denoted $d_{i,j}$ for every $0 \leq i \leq |a|$ and $0 \leq j \leq |b|$. Each node corresponds to a cell of the matrix used to compute the edit distance (see Figure 1). We define four sets of edges called the *horizontal* edges H, the *vertical* edges V, the *diagonal* edges D, and the *equality* edges E_q.

$$\begin{aligned} H &= \{(d_{i,j-1}, d_{i,j}) \mid 0 \leq i \leq |a|, 0 < j \leq |b|\} \\ V &= \{(d_{i-1,j}, d_{i,j}) \mid 0 < i \leq |a|, 0 \leq j \leq |b|\} \\ D &= \{(d_{i-1,j-1}, d_{i,j}) \mid 0 < i \leq |a|, 0 < j \leq |b|\} \\ E_q &= \{(d_{i-1,j-1}, d_{i,j}) \mid 0 < i \leq |a|, 0 < j \leq |b|, \\ &\qquad a_i = b_j\} \end{aligned}$$

Note that the diagonal edges and the equality edges connect the same nodes. The graph allows double edges. However, the equality edge between nodes $d_{i-1,j-1}$ and $d_{i,j}$ is only allowed when $a_i = b_j$. Horizontal, vertical, and diagonal edges have a weight of 1 while equality edges have a weight of 0. Figure 2 shows a part of the graph with incoming edges to node $d_{i,j}$.

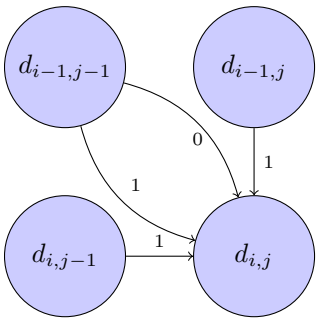


Figure 2: Representation of node $d_{i,j}$ in the edit distance graph and its four ingoing edges.

The length of a path is the sum of the weights of the edges composing the path. The following lemma shows the relation between the length of a path in the graph and the edit distance between two sequences.

Lemma 1. *The edit distance between the sequences a and b is equal to the length of the shortest path between nodes $d_{0,0}$ and $d_{|a|,|b|}$.*

Proof. We want to show that the length of the shortest path from $d_{0,0}$ to each node $d_{i,j}$ of the graph corresponds to the edit distance $d_{i,j}$ between the subsequences $a_1 \dots a_i$ and $b_1 \dots b_j$ as defined in (7). We proceed by induction.

Base case: When $j = 0$, the distance from node $d_{0,0}$ to $d_{i,0}$ is equal to i since only edges in H of weight 1 can be used and i edges of weight 1 are needed to reach node $d_{i,0}$. The same principle applies to the base case when $i = 0$ where only edges of V can be used.

Induction step: We suppose that the length of the shortest path from $d_{0,0}$ to the nodes $d_{i-1,j}$, $d_{i,j-1}$, and $d_{i-1,j-1}$, with $i, j \geq 1$, is equal to the edit distances

between the corresponding subsequences. We prove that the proposition holds for node $d_{i,j}$.

Node $d_{i,j}$ can be reached by four edges: one horizontal, one vertical, one diagonal, and one equality edge. Therefore, the shortest path must pass by one of the nodes $d_{i,j-1}$, $d_{i-1,j}$, and $d_{i-1,j-1}$ and then reach the node $d_{i,j}$ via a single edge.

A shortest path passing by the node $d_{i-1,j-1}$ reaches $d_{i,j}$ with an equality edge if $a_i = b_j$ since this edge has a null weight. Otherwise, the path passes by the diagonal edge of weight 1. In either case, the length of this path is given by $d_{i-1,j-1} + I(a_i \neq b_j)$.

The shortest distance to $d_{i,j}$ is therefore given by $\min(d_{i-1,j}+1, d_{i,j-1}+1, d_{i-1,j-1}+I(a_i \neq b_j))$ which corresponds to the definition of the edit distance given in (7). □

In Figure (1), the cells in bold represent the nodes through which passes the shortest path.

Consider an augmented version of the graph where the equality edges exist even if the relation $a_i = b_j$ does not hold.

$$E'_q = \{(d_{i-1,j-1}, d_{i,j}) \mid 0 < i \leq |a|, 0 < j \leq |b|\}$$

Let E' be the set of edges in the augmented graph.

$$E' = H \cup V \cup D \cup E'_q$$

We model the shortest path problem on the augmented graph and forbids the augmented equality edges to be used whenever $a_i \neq b_j$. Since in every set H, V, D, or E'_q there is at most one in-going edge to a node u, we denote by $f(v, S)$ the binary variable that is set to 1 if and only if the edge in S entering u lies on the shortest path. Let $q_{i,j}$ be a binary variable equals to 1 if and only if $a_i \neq b_j$. We obtain the following model.

Minimize

$$\sum_{\substack{S \in \{H,V,D\} \\ (u,v) \in S}} f(v, S) \tag{9}$$

subject to, for all $1 \leq i \leq |a|$ and $1 \leq j \leq |b|$

$$a_i - b_j \leq |\Sigma| q_{i,j} \tag{10}$$
$$a_i - b_j \geq -|\Sigma| q_{i,j} \tag{11}$$
$$f(d_{i,j}, E'_q) \leq 1 - q_{i,j} \tag{12}$$

for all vertex $u \in V \setminus \{d_{0,0}, d_{|a|,|b|}\}$

$$\sum_{\substack{S \in \{H,V,D,E'_q\} \\ (v,u) \in S}} f(u, S) = \sum_{\substack{S \in \{H,V,D,E'_q\} \\ (u,v) \in S}} f(v, S) \tag{13}$$

and

$$\sum_{S \in \{H,V,D,E'_q\}} f(d_{|a|,|b|}, S) = 1 \tag{14}$$

where

$$a_i, b_j \in \Sigma, q_{ij} \in \{0, 1\}, 0 \leq f(u, v, S) \tag{15}$$

The objective function (9) ensures that we minimize the number of edges with weight one that lie on the shortest path. The constraints (10) and (11) ensures that $q_{i,j} = 1$ whenever $a_i \neq b_j$. Constraint (12) ensures that the equality edges are not chosen when $a_i \neq b_j$. Constraint (13) is the flow conservation constraint that ensures that if the path enters the node u then it needs to leave the node u. The starting node $d_{0,0}$ and the ending node $d_{|a|,|b|}$ are not subject to the flow conservation constraint. Finally, constraint (14) ensures that there is one and only one path ending at node $d_{|a|,|b|}$. Because of the flow conservation constraint, the path has no other choice to start at node $d_{0,0}$.

The character variables a_i and b_j must be integers in Σ. The variables $q_{i,j}$ encoding the differences between two characters must be 0 or 1. Finally, the flow variables $f(u, v, S)$ do not need to be restrained to integers. It is well known (Ahuja, Magnanti & Orlin 1993) that the matrix that encodes the linear constraints of a shortest path problem is totally unimodular and, consequently, the simplex method always returns an integer optimal solution. Consequently, declaring the variables $f(u, v, S)$ as real variables rather than integer variables facilitates the computation made by the MIP solver and reduces the computation times.

In the next section we model the Damerau-Levenshtein distance by building upon the edit distance model.

4 A MIP MODEL THAT MINIMIZES THE DAMERAU-LEVENSHTEIN DISTANCE

In this section, the previous model is extended to minimize the Damerau-Levenshtein distance. We augment the graph by adding the transposition edges.

$$T = \{(d_{i-2,j-2}, d_{i,j}) \mid 2 \leq i \leq |a|, 2 \leq j \leq |b|, a_i = b_{j-1}, a_{i-1} = b_j\}$$

The transposition edges have a weight of one. Figure 3 shows a partial representation of the graph.

Lemma 2. *The Damerau-Levenshtein distance between the sequences a and b is equal to the length of the shortest path between nodes $d_{0,0}$ and $d_{|a|,|b|}$.*

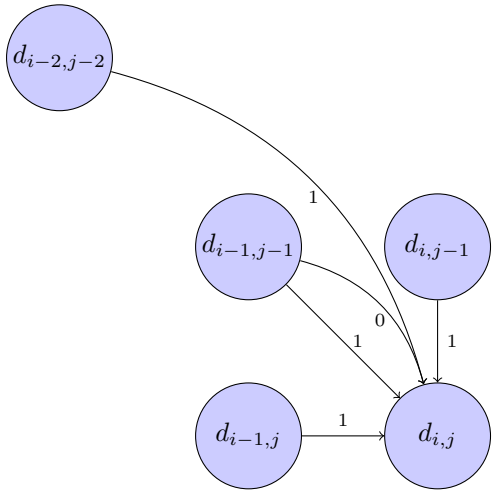


Figure 3: Partial representation of a Damerau-Levenshtein distance graph

Proof. We want to show that the length of the shortest path from $d_{0,0}$ to each node $d_{i,j}$ of the graph corresponds to the Damerau-Levenshtein distance between the subsequences $a_1 \dots a_i$ and $b_1 \dots b_j$ as defined in (8). We proceed by induction.

Base case: If $i, j \leq 1$ then the graph and the Damerau-Levenshtein distance are identical to the ones of edit distance. Lemma 1 applies.

Induction step: We suppose that the length of the shortest path from $d_{0,0}$ to the nodes $d_{i-1,j}$, $d_{i,j-1}$, $d_{i-1,j-1}$, and $d_{i-2,j-2}$, with $i, j > 1$, is equal to Damerau-Levenshtein distances between the corresponding subsequences. We prove that the proposition holds for node $d_{i,j}$.

Node $d_{i,j}$ can be reached by five edges: one horizontal, one vertical, one diagonal, one equality edge, and one transposition edge. Therefore, the shortest path must pass by one of the nodes $d_{i,j-1}$, $d_{i-1,j}$, $d_{i-1,j-1}$, and $d_{i-2,j-2}$ and then reach the node $d_{i,j}$ via a single edge.

A shortest path passing by the node $d_{i-1,j-1}$ reaches $d_{i,j}$ with an equality edge if $a_i = b_j$ since this edge has a null weight. Otherwise, the path passes by the diagonal edge of weight 1. In either case, the length of this path is given by $d_{i-1,j-1} + I(a_i \neq b_j)$.

The shortest distance to $d_{i,j}$, when $a_i = b_{j-1}$ and $a_{i-1} = b_j$ is therefore given by $\min(d_{i-1,j}+1, d_{i,j-1}+1, d_{i-1,j-1} + I(a_i \neq b_j), d_{i-2,j-2} + 1)$ which corresponds to the last case in the definition of the Damerau-Levenshtein distance given in (8). If, however,r $a_i \neq b_{j-1}$ or $a_{i-1} \neq b_j$, there is no transposition edge incident to $d_{i,j}$ and the shortest distance to $d_{i,j}$ is given by $\min(d_{i-1,j}+1, d_{i,j-1}+1, d_{i-1,j-1}+I(a_i \neq b_j))$ which is the third case in equation (8). □

As we did for the edit distance, we consider an augmented version of the graph where the transposition edges exist even when the conditions $a_i = b_{j-1}$ and $a_{i-1} = b_j$ are false.

$$T' = \{(d_{i-2,j-2}, d_{i,j}) \mid 2 \leq i \leq |a|, 2 \leq j \leq |b|\}$$

However, we forbid these edges to lie on the shortest path whenever $a_i = b_{j-1}$ or $a_{i-1} = b_j$ do not hold. We obtain the following model.

Minimize

$$\sum_{\substack{S \in \{H,V,D,T'\} \\ (u,v) \in S}} f(v, S) \tag{16}$$

subject to, for all $1 \leq i \leq |a|$ and $1 \leq j \leq |b|$

$$a_i - b_j \leq |\Sigma| q_{i,j} \tag{17}$$
$$a_i - b_j \geq -|\Sigma| q_{i,j} \tag{18}$$
$$f(d_{i,j}, E'_q) \leq 1 - q_{i,j} \tag{19}$$
$$f(d_{i,j}, T') \leq 1 - q_{i-1,j} \tag{20}$$
$$f(d_{i,j}, T') \leq 1 - q_{i,j-1} \tag{21}$$

for all vertex $u \in V \setminus \{d_{0,0}, d_{|a|,|b|}\}$

$$\sum_{\substack{S \in \{H,V,D,E'_q,T'\} \\ (v,u) \in S}} f(u, S) = \sum_{\substack{S \in \{H,V,D,E'_q,T'\} \\ (u,v) \in S}} f(v, S) \tag{22}$$

and

$$\sum_{S \in \{H,V,D,E'_q,T'\}} f(d_{|a|,|b|}, S) = 1 \tag{23}$$

where

$$a_i, b_j \in \Sigma, q_{ij} \in \{0, 1\}, 0 \leq f(u, v, S) \tag{24}$$

Constraints (20) and (21) ensure that the transposition edges do not lie on the shortest path when the conditions $a_i = b_{j-1}$ or $a_{i-1} = b_j$ are false. The transposition edges were added to the summations in the objective function, in the flow conservation constraints, and in constraint (23) that guarantees a unique path.

5 EXPERIMENTS

We carried out experiments with industrial data for a combined planning and scheduling problem from the forest-products industry described in (Gaudreault, Forget, Frayret, Rousseau, Lemieux & D'Amours 2010).

In a lumber finishing facility, lumbers are first planed (or surfaced). They are then sorted according to their

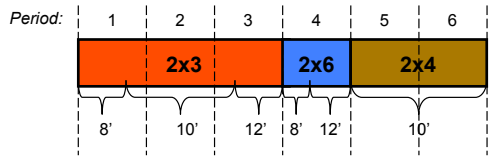


Figure 4: Production plan for a finishing line for six consecutive production shifts (periods).

grade (i.e. quality) with respect to the residual moisture content and physical defects. Lumber may be trimmed in order to produce a shorter lumber of a higher grade and value. This process is usually optimized by hardware to produce products with the highest value, with no consideration for the actual customer demand. This causes the production of multiple product types at the same time (co-production) from a single product type in input (divergence). It is important to note that the co-production cannot be avoided from a planning point of view: it is embedded within the transformation process. It is common to obtain more than 20 different types of products from a single product. There is a setup cost each time the facility processes a different family of products (e.g. going from 2"x3" to 2"x6") but no setup cost to go from one length of product to another (e.g. 2"x3"x8' to 2"x3"x7'). Consequently, most sawmills allow such a setup only between production shifts as they prefer campaigns (a batch of products of the same dimension but variable length) with a duration of more than one shift. To sum up, the decisions that must be taken in order to plan the finishing operations are the following: (1) which campaign to realize (i.e. which lumber dimensions), (2) when and for how long and (3) for each campaign, what quantities of each length to process.

Figure 4 shows a simple example of a production plan, including the campaigns (2"x3", 2"x6" and 2"x4") and the time spent on each length.

When changing the production plan, the industry prefers to minimize the number of times that a period is associated to a different product family. This is more important than minimizing the number of changes of setup times. This is due to the additional work needed to reorganize the shifts when the product-family changes. Setup costs are considered non significative when compared to new revenues generated by the new orders Accepting new orders also greatly reduces the potential new setup costs by adding new revenues. The optimization does not support changes to the supply of raw material since it depends on the step of drying lumber which can last from a few days to months depending on the method used.

5.1 Methodology

As a mixed integer program (MIP) is known to solve the wood-finishing operations, we suppose that a production plan is in place (we call it the *original* production plan).

We then simulate new orders arrivals. For each order, the due date is chosen over all periods of the planning horizon using a uniform distribution. The ordered products is chosen using a multi-nomial distribution fitted on the datasets. The ordered quantity is modeled as a normal distribution.

When new orders arrive, we try to come up with a new plan satisfying this new order with as less lateness as possible. This is done using the original MIP model, but with an additional constraint preventing it from increasing lateness for already accepted orders. We then measure the Hamming distance, edit distance and Damerau-Levenshtein of this new plan by looking at the changes in the product-family consumed at each period.

As a final step, we try to find an equivalent plan (no more lateness for each accepted order) that minimizes the distance with the original plan. This can be done using any of the approaches described in previous sections (minimizing the Hamming, edit or Damerau-Levenshtein distance).

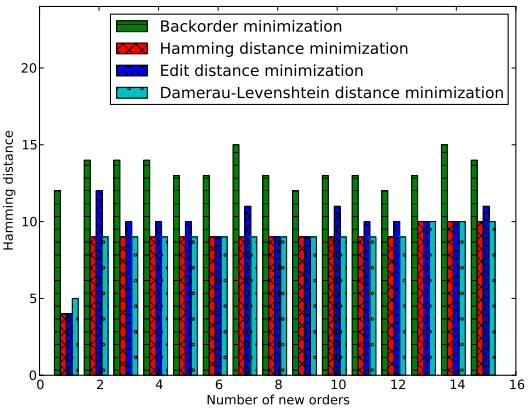
5.2 Results

We worked with two instances of the problem (M1 and M2) provided by a Canadian wood products company. Instance M1 is an easy instance, while M2 is larger and more difficult. The models are solved with CPLEX 12.5 on a computer using eight Intel Core i7-2600 CPU 3.40GHz and 4GB ram.

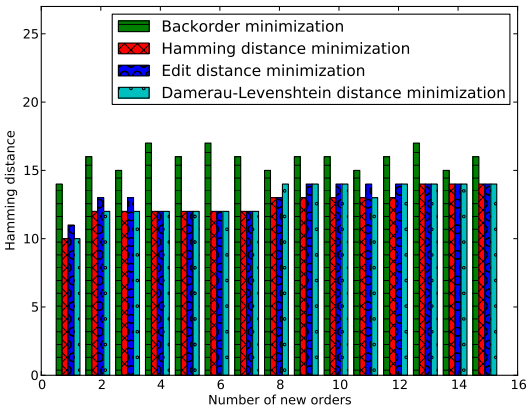
Figures 5a to 5c show the results obtained for the instance M1. For each metric, they show the distance between the original plan and the new one according to the model that was used for replanning (the original MIP model, the model minimizing the Hamming

	Backorders minimization	Hamming distance minimization	Edit distance minimization	Damerau-Levenshtein distance minimization
M1	1.4	0.7	4.7	4.2
M2	564.5	7.9	135.8	127.3

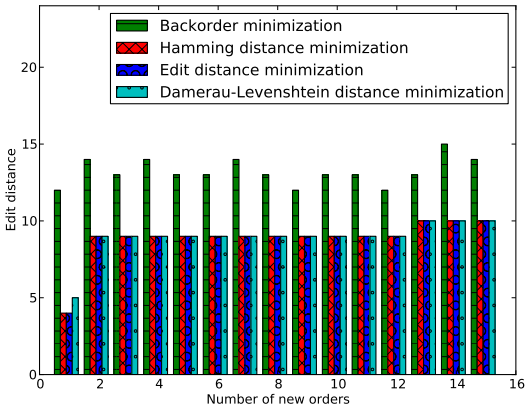
Table 1: Geometric mean of the run times (seconds) for each experiment.



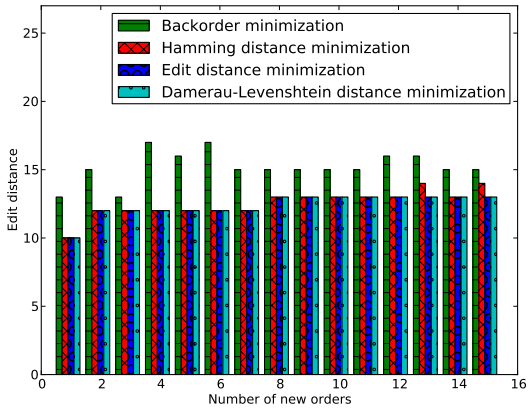
(a) Hamming distance for dataset M1 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.



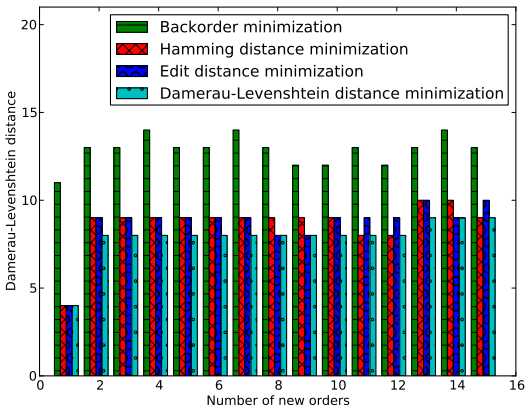
(d) Hamming distance for dataset M2 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.



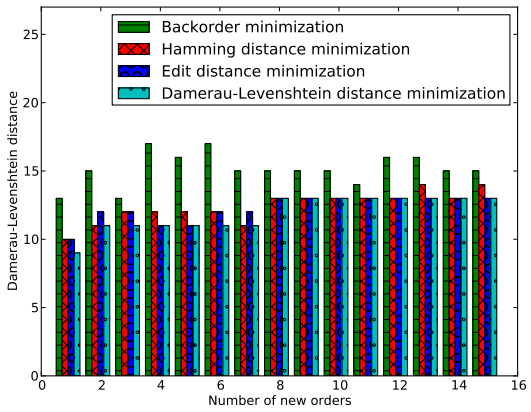
(b) Edit distance for dataset M1 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.



(e) Edit distance for dataset M2 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.



(c) Damerau-Levenshtein distance for dataset M1 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.



(f) Damerau-Levenshtein distance for dataset M2 when minimizing backorders, hamming distance, edit distance and Damerau-Levenshtein distance.

Figure 5: Distances obtained between the old and new plans after minimizing different metrics.

distance, the model minimizing the edit distance, or the model minimizing the Damerau-Levenshtein distance). Figures 5d to 5f shows results for the instance M2.

The Damerau-Levenshtein distance being a more refined metric than the other one (it would be the one a company would like to use) our analysis focuses on Figures 5c and 5f.

Fist of all, it appears the number of new orders we accumulate before we decide to replan does not have a strong impact on how far the plan is perturbed. Second, the three models we propose provide much less perturbation than just using the original model the minimizing order lateness (backorders).

Since the Hamming distance is an upper bound on the edit and Damerau-Levenshtein distance, the Hamming distance minimization always leads to a greater or equal edit and Damerau-Levenshtein distance. However, in both edit distance and Damerau-Levenshtein distance minimization, there are multiple cases where the distances are improved over the Hamming distance minimization.

We can see from Table 1 that reoptimizing a production plan in order to minimize the Hamming distance can be done in a very short time (in comparison to the time needed to first minimize lateness for the new order). For a production plan of length n, only n new variables are added to the model so this approach has a small overhead.

The edit distance model takes more time to solve than the Hamming distance. This is due to the greater number of variables added. For a production plan of length n, $4n^2$ new variables are added to the model.

The time needed to obtain an optimal solution for the Damerau-Levenshtein model is slightly lower on average than the time needed by the edit distance models as shown in Table 1 even if more decision variables are added. More precisely, $4n^2 + (n-2)^2$ new variables are added for a production plan of length n. This lower runtime is due to the possibility of obtaining a lower bound earlier in the search due to the additional edges which represent the transposition of adjacent elements.

However, runtime for the three distance-minimizing models are quite smaller in comparison to the original order-lateness minimization model. This shows that our approach has a small cost when compared to only minimizing the order-lateness (backorders).

6 CONCLUSION

We proposed a new technique for replanning and scheduling while minimizing perturbation. With this approach, the production plans are represented as sequences and the objective function is to minimize the distance between the previous plan and the new plan. Three linear models are presented for three different distance metrics: the Hamming distance, the edit distance, and the Damerau-Levenshtein distance. We successfully applied this technique on the planning and scheduling of wood finishing operations. Empirical results show that the solution does replan the operations with a minimum of perturbation while keeping the computation time short. The choice of the best distance metric to use is highly dependant on the industry specifications and should be studied in future work.

References

Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993). Network flows: theory, algorithms, and applications.

Bixby, A., Downs, B. & Self, M. (2006). A scheduling and capable-to-promise application for swift & company, *Interfaces* **36**(1): 69–86.

Brown, G. G., Cormican, K. J., Lawphongpanich, S. & Widdis, D. B. (1997). Optimizing submarine berthing with a persistence incentive, *Technical report*, DTIC Document.

Brown, G. G., Dell, R. F. & Wood, R. K. (1997). Optimization and persistence, *Interfaces* **27**(5): 15–37.

Chen, J.-H. & Chen, C.-T. (2009). Using mathematical programming on two-phase order promising process with optimized available-to-promise allocation planning, *International Journal of the Computer, the Internet and Management* **17**(3): 25–40.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors, *Communications of the ACM* **7**(3): 171–176.

Fox, M., Gerevini, A., Long, D. & Serina, I. (2006). Plan stability: Replanning versus plan repair., *ICAPS*, pp. 212–221.

Gaudreault, J., Forget, P., Frayret, J.-M., Rousseau, A., Lemieux, S. & D'Amours, S. (2010). Distributed operations planning in the lumber supply chain: Models and coordination., *International Journal of Industrial Engineering: Theory, Applications and Practice* **17**(3).

Hamming, R. W. (1950). Error detecting and error correcting codes, *Bell System technical journal* **29**(2): 147–160.

Kukich, K. (1992). Techniques for automatically correcting words in text, *ACM Computing Surveys (CSUR)* **24**(4): 377–439.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals, *Soviet physics doklady*, Vol. 10, p. 707.

Linus, S. (1991). Lindo: An optimization modeling system, *The Scientific Press, San Francisco* .

Sankoff, D. & Kruskal, J. B. (1983). Time warps, string edits, and macromolecules: the theory and practice of sequence comparison, *Reading: Addison-Wesley Publication, 1983, edited by Sankoff, David; Kruskal, Joseph B.* **1**.