# Towards Optimal Formwork Pairing on Construction Sites

Thierry Benoist

Bouygues e-lab,
32 av Hoche, 75008 Paris
tbenoist@bouygues.com

**Abstract.** Minimizing shutterings assembling time on construction sites can yield significant savings in labor costs and crane moves. It requires solving a pairing problem that optimizes the ability for the crane to move chains of shutterings as a whole when they can be later reused together to frame another wall of the site. In this paper, we show that this problem is NP-hard in the strong sense as well as both its multiflow and ordering aspects. We also introduce a linear relaxation that computes reasonably good lower bounds of the objective, and describe a Tabu Search based on pairings insertion and ejection that builds promising solutions.

**Keywords.** Pairing, Russian Dolls, Tabu, Fixed-Charge Multi-Commodity Flow

## 1. Introduction

A piece of formwork (or shuttering) is a pair of vertical metallic panes used to build walls: concrete is to be poured in between these panes and requires typically one drying day before this shuttering can be removed and reused elsewhere on the site. Different formwork families (characterized by their lengths: typically from 60 to 480cm) are used on a construction site but walls are usually longer than the maximal length: therefore covering a wall requires *assembling* several formworks together (see **Fig. 1**, page 3). For instance a wall of 9.50m can be covered in various ways using formworks of 1,2,3 or 4 meters: 2 x 4m + 1 x 2m ,or 3 x 3m + 1 x 1m, or 1 x 4m + 2 x 3m, etc… For each wall, the main constraint states that the sum of the lengths of allocated formworks must exceed the length of the wall (covering constraint).

Framing walls on a construction site is one of the main tasks of the carcass work (the other is framing floors). A building deck in a residential building like those erected by *Habitat,* a trademark of *BOUYGUES Construction,* is typically made of a hundred walls and completed in around ten days, using a limited number of vertical formworks whose rotation and assembling are planned by the Method Department.

Assembling two shutterings together takes several minutes and requires the most critical resource: the crane. This task can be repeated more than 1000 times per site, thus minimizing the number of junctions to be performed is of great importance. A solution to achieve this goal is to move chains of shutterings from one wall to another without disassembling them, what requires solving a FORMWORK PAIRING problem (FPP) maximizing the size and number of such chains. To our knowledge this problem has never been addressed in the literature despite its impact on labor costs and crane moves.

We first define the problem in the next section and prove its NP-hardness in Section 3. Section 4 proposes a relaxation modeled by a linear program, whose optimal integer solutions are found using a Russian-Doll algorithm. Section 5 is a parenthesis on the ordering aspect of the problem and its complexity. Finally Section 6 describes a greedy algorithm and a Tabu Search approach, whose results are discussed in Section 7.

## 2. The FORMWORK PAIRING PROBLEM

### 2.1. Data: Schedule and Allocation

The first well-known problem arising in the planning of construction sites consists in *scheduling* framing tasks (one per wall), subject to precedence and exclusion constraints. The second problem consists in *allocating* formworks to each wall, satisfying the covering constraint exposed in introduction as well as a lot of side constraints and preferences and tending to minimize the total number of formworks to command for the site (the stock of the site).

The solutions of these problems (walls scheduling and formworks allocation) are the *input* of the FORMWORK PAIRING PROBLEM studied in this paper. We denote by $N$ the number of walls, by $T$ the number of days (makespan) and by $K$ the number of formwork types. The result of the scheduling problem is a function $d:[1,N] \rightarrow [1,T]$ specifying for each wall $w$ the day $d(w)$ it has been assigned to. We denote by $W_t$ the set of walls framed on day $t$: $W_t=\{w\in[1,N] \mid d(w)=t\}$. The set of formworks allocated to wall $w$ is a vector $a_w$ of length $K$ such that $a_w[k]$ is the number of formworks of type $k$ allocated to wall $w$, $|a_w|$ denotes the total number of formworks allocated to wall $w$ ($|a_w| = \sum_k a_w[k]$). Finally, the stock $A[k]$ is the maximum number of formworks of type $k$ that will be used simultaneously on this site: $\forall k\in[1,K]$, $A[k] = max_{t\in[1,T]}\{\sum_{w\in W_t} a_w[k]\}$.

### 2.2. Pairing Model

Given a schedule for the site and an allocated set of formworks for each wall, the FORMWORK PAIRING PROBLEM deals with the ordering of these formworks. Since formworks only differ by their length, they can be assembled in any order. However, as explained in section 1, our goal is to move formwork chains from one wall to another without disassembling them in order to save labor costs and crane moves. Since only consecutive groups can be extracted from a wall assembly to be reused elsewhere, we must arrange allocated formworks in an appropriate order on each wall. For instance both leftmost formworks of **Fig. 1** have been extracted together from a previous wall, thus their junction can be preserved, thus the number of junctions to be performed on this wall is two instead of three.

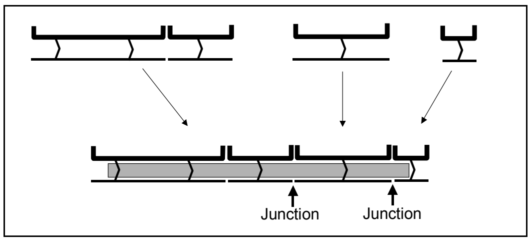


**Fig. 1.** Covering a wall with an assembly of formworks (view from the top)

Formworks allocated to a wall $w$ must be arranged in a sequence of length $|a_w|$. We define the position $(w,p)$ as the $p$th element of this sequence[1]. Hence, the successive utilizations of one formwork can be expressed as a list of positions $\{(w_1,p_1), (w_2,p_2), \ldots\}$[2]. Our goal is to compute these successive positions for each formwork in order to minimize on each wall the number of neighbors in its covering sequence that were not already neighbors on their previous positions. The cost function to be minimized is the total number of these *junctions*.

This description can be formalized into a constrained multi-commodity flow problem, associating a commodity with each of the $K$ formwork types. Thus a directed graph $G(V,E)$ can be defined representing each position $(w,p)$ by a node. We add a source node and a sink node, respectively named (0,1) and $(N+1,1)$. An edge $(w_1,p_1)\rightarrow(w_2,p_2)$ of capacity 1 is defined for each pair of positions such that $d(w_1) < d(w_2)$ (by convention $d(0)=0$ and $d(N+1)=T+1$) This edge models the possibility to reuse the $p_1^{th}$ formwork of wall $w_1$ at the $p_2^{th}$ position of wall $w_2$.

$$V = \left\{(w,p)\,\middle|\, w \in [1,N],\ p \in [1,|a_w|]\right\} \cup \left\{(0,1),(N+1,1)\right\} \tag{1}$$

$$E = \left\{(w_1,p_1) \rightarrow (w_2,p_2) \;\middle|\; (w_1,p_1),(w_2,p_2) \in V^2 \wedge d(w_1) < d(w_2)\right\} \tag{2}$$

In this graph, the successive utilizations of a piece of formwork form a path. Hence a solution of this problem is a flow on $G(V,E)$. One junction is saved each time a pair of parallel edges $\{(w_1,p_1)\rightarrow(w_2,p_2), (w_1,p_1+1)\rightarrow(w_2,p_2+1)\}$ is used. Indeed the use of this pair means that the $p_1$th and $(p_1+1)$th formworks of wall $w_1$ will by moved together to positions $p_2$ and $p_2+1$ of wall $w_2$.

## 2.3. Variables, Cost Function and Constraints

We use two families of variables. The flow function itself is denoted by $X{:}E\times[1,K]\rightarrow Z^+$. We also define a binary variable $Y{:}P\rightarrow\{0,1\}$ on each edge $(w_1,p_1)\rightarrow(w_2,p_2)$ of the following subset $P\subseteq E$:

[1] Both faces of a wall are not identical, therefore an non ambiguous "left to right" indexing can be chosen.

[2] This list codes for "this formwork is first used on the $p_1^{th}$ position of wall $w_1$, then on the $p_2^{th}$ position of wall $w_2$, ... "

$$P = \left\{ (w_1, p_1) \rightarrow (w_2, p_2) \in E \;\middle|\; p_1 < |a_{w_1}| \wedge p_2 < |a_{w_2}| \right\} \qquad \textbf{(3)}$$

This subset $P$ is the set of edges such that the parallel pair $\{(w_1,p_1)\rightarrow(w_2,p_2), (w_1,p_1+1)\rightarrow(w_2,p_2+1)\}$ exists. The use of this pair of parallel edge e.g. the saving of one junction is modeled by $Y_{(w1,p1)\rightarrow(w2,p2)}=1$. Without these savings, the number of junctions to be performed on each wall $w$ would be $|a_w|$-1. Therefore the total number of junctions to be performed on this site reads:

$$\sum_{w\in[1,N]} |a_w| - N - \sum_{(w_1,p_1)\rightarrow(w_2,p_2)\in P} Y_{(w_1,p_1)\rightarrow(w_2,p_2)} \qquad \textbf{(4)}$$

Equation (**4**) is the cost function to be minimized when solving the FPP.

For each commodity $k\leq K$, the flow issued from the source and collected by the sink is $A[k]$ (equation **(5)**). All edges have unitary capacities: more precisely the flow across each node must equal 1 (equation (**6**)) since for each position $(w,p)$ there is only one $p^{th}$ formwork on wall $w$. Besides the flow crossing all nodes $(w,p)$ of a wall $w$ must match the allocation vector of this wall (equation (**7**)): for each commodity $k$ the number of nodes of the wall crossed by one unit of this commodity must equal the number of formworks of this type $k$ allocated to this wall ($a_w[k]$). Finally a pair of parallel edges is used only if both edges are crossed by one unit of flow of any commodity (equation (8)).

$$\forall k \leq K, \sum_{(0,1)\rightarrow(w,p)\in E} X^{(k)}_{(0,1)\rightarrow(w,p)} = A[k] \;\; and \sum_{(w,p)\rightarrow(N+1,1)\in E} X^{(k)}_{(0,1)\rightarrow(w,p)} = A[k] \qquad \textbf{(5)}$$

$$\forall (w,p) \in V \setminus \{(0,1),(N+1,1)\}, \begin{cases} \displaystyle\sum_{(w_1,p_1)\rightarrow(w,p)\in E} \sum_{k\leq K} X^{(k)}_{(w_1,p_1)\rightarrow(w,p)} = 1 \\ \displaystyle\sum_{(w,p)\rightarrow(w_2,p_2)\in E} \sum_{k\leq K} X^{(k)}_{(w,p)\rightarrow(w_2,p_2)} = 1 \end{cases} \qquad \textbf{(6)}$$

$$\forall w \in [1,N], \forall k \leq K, \sum_{p\leq n_w} \sum_{(w_1,p_1)\rightarrow(w,p)\in E} X^{(k)}_{(w_1,p_1)\rightarrow(w,p)} = a_w[k] = \sum_{p\leq n_w} \sum_{(w,p)\rightarrow(w_2,p_2)\in E} X^{(k)}_{(w,p)\rightarrow(w_2,p_2)} \qquad \textbf{(7)}$$

$$\forall (w_1,p_1)\rightarrow(w_2,p_2) \in P, \; Y_{(w_1,p_1)\rightarrow(w_2,p_2)} \leq \frac{1}{2} \sum_{k\leq K} \left( X^{(k)}_{(w_1,p_1)\rightarrow(w_2,p_2)} + X^{(k)}_{(w_1,p_1+1)\rightarrow(w_2,p_2+1)} \right) \qquad (8)$$

## 2.4. Example

**Fig. 2** shows an optimal flow (saving 11 junctions) for a problem where 10 walls must be framed in 5 days using formworks of 6 families. In this picture:

- Edges toward the sink and unused edges have not been drawn, for the sake of readability.

- Sets of parallel edges have been represented by single bold edges labeled by the number of formworks moved as a whole, i.e. the number of saved junctions plus one.
- Nodes have been grouped by wall and labeled by the commodity of the unit of flow crossing them. Consider for instance wall 4 (the only one framed on Wednesday): its 1st formwork (position (4,1)) has type 4, the second one (position (4,2)) has type 1, and so on…

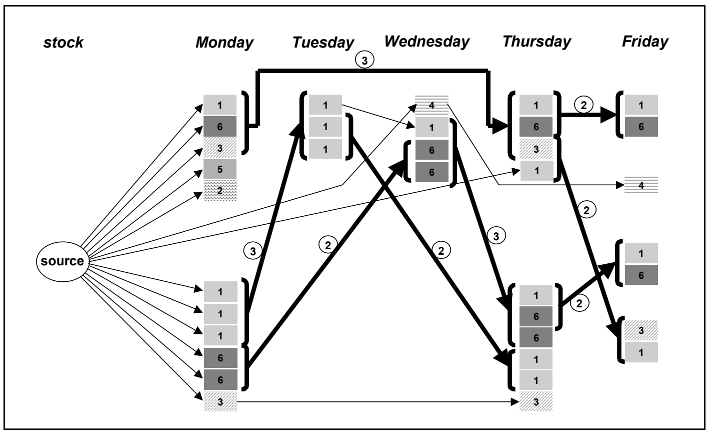


**Fig. 2.** A 10-walls example

## 3. NP-hardness

In order to prove the NP-hardness of the FORMWORK PAIRING PROBLEM, we will prove the NP-completeness of the corresponding decision problem (belonging to NP).

### 3.1. Reduction to CONSTRAINED MINIMUM EDGE-COST FLOW

Let us consider the specific one-family case (K=1). In this case, shutterings can be freely ordered since all of them are identical. Thus the number of junctions on each wall is the "number of its sources" minus one since the concatenation of incoming formwork chains is feasible and is the best ordering. This leads to a flow problem (in a layered graph such that each node corresponds to a wall, and each layer represents a working day) where the number of active incoming edges must be minimized at each node, i.e. the total number of used edges (on which the flow is strictly greater than zero) must be minimized (**11**). More precisely the decision problem can be reformulated as follows, aggregating positions of each wall $w$ into a single node crossed by a flow $|a_w|$ (**10**).

NAME: CONSTRAINED MINIMUM EDGE-COST FLOW

INSTANCE: A directed graph $G = (V,E)$ whose vertices can be partionned into layers $V = \{s\} \cup V_1 \cup V_2$ ….. $\cup V_T \cup \{t\}$ such that $E=\{(w_1 \rightarrow w_2) \mid w_1=s \vee w_2=t \vee \exists\ i<j, w_1 \in V_i\ w_2 \in V_j\}$, a demand function $a: V \backslash \{s,t\} \rightarrow Z^+$ and two positive integers $A$ and $B$.

QUESTION: Is there a flow function $X{:}A\rightarrow Z^{+}$ such that:

$$\sum_{(s\rightarrow w)\in E} X_{s\rightarrow w} \le A, \tag{9}$$

$$\forall w \in V \setminus \{s,t\}, \sum_{(w'\rightarrow w)\in E} X_{w'\rightarrow w} = \sum_{(w\rightarrow w'')\in E} X_{w\rightarrow w''} = a_w, \tag{10}$$

$$\text{and } |\{(w_1, w_2)\in E \mid X_{w1\rightarrow w2}>0\}| \le \mathrm{B} \quad ? \tag{11}$$

Finally the one-family case is a variant of the MINIMUM EDGE-COST FLOW problem[3] (proved NP-complete in [5]) on a *layered* graph where the flow across each node is *fixed*. To our knowledge this variant is not known to be NP-complete[4].

### 3.2. Complexity proof

We will prove the NP-completeness of the BIPARTITE MINIMUM EDGE-COST FLOW problem defined below, which is a sub case of CONSTRAINED MINIMUM EDGE-COST FLOW.

NAME: BIPARTITE MINIMUM EDGE-COST FLOW

INSTANCE: A bipartite graph G = (U∪V, A), a demand function $a{:}\mathrm{U}\cup\mathrm{V}\rightarrow \mathrm{Z}^{+}$ and a positive integer *B*.

QUESTION: Is there a flow function $X{:}A\rightarrow Z^{+}$ such that:

$$\forall u \in U \ \sum_{v\in V} X_{u\rightarrow v} = a_u \ , \quad \forall v \in V \ \sum_{u\in U} X_{u\rightarrow v} = a_v \quad \left|\{(u\rightarrow v)\in E,\ X_{u\rightarrow v}>0\}\right| \le B \tag{12}$$

**Proposition 1.** *BIPARTITE MINIMUM EDGE-COST FLOW is NP-complete in the strong sense.*

**Proof.** Let us consider the NP-complete 3-PARTITION problem [4]:

INSTANCE: A finite set X of *3m* positively weighted elements such that:

$$\forall x \in X,\ \frac{3\overline{w}}{4} < w(x) < \frac{3\overline{w}}{2} \tag{13}$$

where *w(x)* is the weight of *x* and $\overline{w}$ is the average weight.

QUESTION: Can X be partitioned into *m* disjoint subsets of equal weight $3\overline{w}$? (note that equation (**13**) enforces these subsets to contain exactly three elements).

With any instance of 3-PARTITION, one can associate an instance of BIPARTITE MINIMUM EDGE-COST FLOW by the following polynomial transformation.

---

[3] MINIMUM EDGE-COST FLOW is also named FIXED CHARGE FLOW.

[4] The classical transformation from X3C suggested in [5] does not address the complexity of this variant because it makes use of one node per 3-set, that is selected if and only if a non-null flow crosses the corresponding node. This mechanism is obviously not operational when flows across all nodes are fixed.

Each $x \in$ X is associated with a left vertex $u$ source of $w(x)$ units of flow. We create $m$ right vertices each of which is a sink of $3\overline{w}$ units of flow, and consider the complete bipartite graph G with infinite capacities on all edges, searching for a flow in this graph with less than $3m$ active edges.

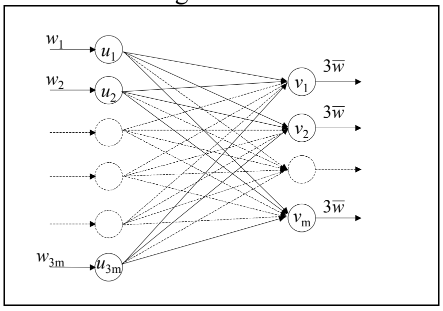


**Fig. 3.**Transformation from 3-PARTITION

$\Rightarrow$: Assume $X_1, X_2, \ldots X_m$ is a balanced partition of X (each $X_i$ weighting $3\overline{w}$). If we name $U_1, U_2, \ldots U_m$ the corresponding subsets of left vertices, the flow using only edges $\{(u,v_i), \forall\ i \leq m, \forall\ u \in U_i\}$ is trivially feasible and only activates $3m$ edges (one per left vertex).

$\Leftarrow$: Assume reciprocally that $F$ is a flow on G activating only $3m$ edges. Since no weight is null, we can infer that for each left vertex $u$ there is exactly one active edge $(u,v_i)$. If for each $i$ we name $U_i$ the set of left vertices $u$ such that $(u,v_i)$ is active, then the corresponding subsets $X_1, X_2, \ldots X_m$ are a balanced partition of X.

Finally BIPARTITE MINIMUM EDGE-COST FLOW is NP-complete in the strong sense as soon as the number of right vertices is greater than three. It remains NP-complete at least in the weak sense with only two right vertices (analogous transformation from SUBSET SUM [2]). ■

Proposition 1 proves that the one-family case of the FPP is NP-hard in the strong sense even in a bipartite graph. Thus the general case (K>1) is NP-hard too.

## 4. Linear relaxation of the FPP

The linear model presented in section 2 contains $O(K \times (\Sigma|a_w|)^2)$ variables[5] with highly discrete constraints, making this MIP not tractable with Xpress-MP for instance. Therefore we introduce in this section a simpler cost function (lower bound of the number of junctions to be performed) that can be minimized in a lighter model, namely a FIXED CHARGE MULTI-COMMODITY FLOW [7] where the flow across each node is fixed.

Intuitively, this relaxation relies on an argument evoked in section 3.1: if formworks used to cover a wall $w$ come from $Z$ different walls, then at least $Z$-$1$ junctions must be performed on wall $w$. Therefore the objective of this relaxation is to minimize the number of active $w_1 \rightarrow w_2$ links.

[5] Which means up to one million binary variables on practical instances.

Although NP-hard, we will prove that this relaxation is usually tractable for real size problems and yields good lower bounds.

### 4.1. Relaxation

Let us add a new set of Boolean variables to the model defined in section 2, detecting for each pair of walls whether formworks are moved from the first wall to the second wall or not (noting $\Omega_{<}=\{w_1,w_2\in\Omega^2 \mid d(w_1)<d(w_2)\}$ for any $\Omega\subseteq[0,N+1]$).

$$Z:[1,N]_{<}\mapsto\{0,1\}, Z_{w_1\to w_2}=1 \Leftrightarrow X_{w_1\to w_2}\geq 1 \quad \textit{where } \forall w_1,w_2\in[0,N+1]_{<}\ X_{w_1\to w_2}=\sum_{k\leq K}\sum_{\substack{p_1\leq|a_{w_1}|\\ p_2\leq|a_{w_2}|}} X^{(k)}_{(w_1,p_1)\to(w_2,p_2)} \tag{14}$$

**Proposition 2.** For all solutions of this enriched model:

$$\sum_{k\leq K}A[k]+\sum_{w_1,w_2\in[1,N]_{<}}Z_{w_1\to w_2}-N\leq\sum_{w\in[1,N]}|a_w|-N-\sum_{(w_1,p_1)\to(w_2,p_2)\in P}Y_{(w_1,p_1)\to(w_2,p_2)} \tag{15}$$

**Proof.** Since $\Sigma_k A[k]=\Sigma_w X_{0\to w}$ (equation **(5)**), the difference between the left hand side and the right hand side of (**15**) can be rewritten:

$$\sum_{w\in[1,N]}\left(X_{0\to w}+\sum_{\substack{w_0\in[1,N]\\ d(w_0)<d(w)}}\left(Z_{w_0\to w}+\sum_{\substack{p<|a_{w_0}|\\ q<|a_w|}}Y_{(w_0,p)\to(w,q)}\right)-|a_w|\right) \tag{16}$$

For each $w_0$ such that $d(w_0)<d(w)$, $Z_{w0\to w}$ equals either:

- 0 in which case the corresponding $\Sigma y$ term is null too,
- or 1, in which case at most $X_{w0\to w}-1$ pairs of parallel arcs are activated between $w_0$ and $w$, where $X_{w0\to w}$ is the aggregated flow from nodes of $w_0$ to nodes of $w$.

Since $|a_w|$ is also the sum of $X_{w0\to w}$ on all $w_0$ preceding $w$ (including 0), (**16**) is always negative. ■

### 4.2. Linear model

The left hand side of equation (**15**) is not only a lower bound of the original objective: it also allows simplifying the model since positions and parallel edges are irrelevant to this relaxation. In other words it is possible to aggregate position nodes of each wall $w$ in a single node, crossed by $a_w[k]$ units of each commodity. Since the objective is to minimize the number of activated edges (we omit the $\Sigma A[k]$–N constant term in what follows), this relaxed problem is a FIXED CHARGE MULTI-COMMODITY FLOW [7] (NP-hard). In this new graph $G'=(V',E')$, where $V'=[0,N+1]$ and $E'=\{w_1\to w_2 \mid w_1,w_2\in V'^2 \wedge d(w_1)<d(w_2)\}$, the objective is to minimize the sum of $Z$ variables subject to three families of constraints:

$$\forall k \le K, \qquad \sum_{0\to w\in E'} X^{(k)}_{0\to w} = A[k] \tag{17}$$

$$\forall w \in [1,N], \forall\, k \le K, \sum_{w_1\to w\in E'} X^{(k)}_{w_1\to w} = a_w[k] = \sum_{w\to w_2\in E'} X^{(k)}_{w\to w_2} \tag{18}$$

$$\forall w_1 \to w_2 \in E', \forall\, k \le K, X^{(k)}_{w_1\to w_2} \le \min(a_{w_1}[k], a_{w_2}[k])) \times Z_{w_1\to w_2} \tag{19}$$

In the remaining of the paper, this relaxation will be referred to as *R*(FPP). In practice, it can be interpreted as a focus on the minimization of the number of crane moves, independently of assembling times. It determines the sequence of walls covered by each formwork and provides a lower bound. However, given a solution of the relaxed problem *R*(FPP), deciding whether these shutterings can be ordered to produce an FPP solution of equal cost will be proved NP-complete in section 5.

## 4.3. Valid inequalities

The above model can be enriched with classical valid inequalities concerning incoming edges of each node (and symmetrical cuts can be posted on outgoing edges). For each (non-sink) node *w* of *V'* we define $E_w \subset E'$, the set of incident arcs of *w*. A subset $C\subset E_w$ is said to *cover w* (noted $C \succ w$) when: $\forall\ k\le K,\ \ \Sigma_{v\to w\in C}\, a_v[k] \ge a_w[k]$. Thus $E_w$ can be partitioned into $E_w^+ \cup E_w^-$, where $E_w^+$ is the set of covering edges (any singleton of $E_w^+$ covers *w*). From now on, for any subset *C* of *E'* we note $\Sigma(C)$ the number of actives arcs in this subset i.e. $\Sigma(C) = \Sigma_C Z$.

When $E_w^- = \varnothing$, the only covering cut would be $\Sigma(E_w) \ge 1$ but it is always redundant since $\forall\ k \le K$, $\Sigma(E_w) \ge \Sigma X^{(k)}/_{aw[k]}$ is induced by (**18**) and (**19**). Otherwise (when $E_w^- \ne \varnothing$), two types of valid inequalities can be generated:

- no subset whose complementary does not cover *w* can be totally unused i.e. if $E_w^- \backslash C$ does not cover *w*, $\Sigma(\, E_w^+ \cup C) \ge 1$ is valid.
- if $E_w^-$ contains no covering subset of cardinality strictly smaller than some $\alpha$, then when no edge of $E_w^+$ is used, at least $\alpha$ edges of $E_w^-$ need to be activated, thus $\Sigma(E_w^-) \ge \alpha(1 - \Sigma(E_w^+))$ is valid.

Even after adding these cuts for all $C \subset E_w^-$, $|C| \le 2$, 24 hours CPU is not sufficient to complete the search (with XPRESS-MP) on real-size problems (around 100 walls): integral solutions are found quickly but their optimality is not proven. This poor result is not completely surprising since flow cover cuts are known [6] to be usually not facet defining when not lifted. Instead of lifting valid inequalities, we propose in the next section a simple iterative procedure adapted to the layered structure of the graph.

## 4.4. Russian Dolls Search

The Russian Dolls algorithm [8] was first developed to solve an Earth Observation Satellite Scheduling problem [1] by constraint optimization techniques. The idea is to successively solve growing nested sub-problems, starting by the schedule of the very last tasks (photos) and ending by solving the whole problem. Each sub-problem provides a good bound boosting the resolution of the next ones, what makes the whole process much faster than a direct resolution of the whole problem.

What we propose here is to adapt this mechanism to solve our Integer Linear Program ((**17**),(**18**),(**19**)). Given any subset of edges $C\subset E'$, a lower bound of $\Sigma(C)$ can be obtained solving the linear program with objective $\Sigma(C)$, and relaxing all integrality constraints on edges of $E'\backslash C$; we note this lower bound $LB(C)$. Thus the search applies on an ordered collection $\{C_1, C_2,\ldots C_m\}$ of subsets of $E'$, with $C_m=E'$: for each subset $C_i$ it computes $LB(C_i)$ and posts $\Sigma(C_i)\geq LB(C_i)$. Finally $LB(C_m)=LB(E')$ is the optimal integer solution.

Because of the layered structure of the graph, it seems natural to consider the following subsets (this interval reasoning can be compared to [3]):

$$\forall t_1,t_2 \leq T,\ [t_1,t_2]=\left\{w_1 \rightarrow w_2 \in E' \,\middle|\, d(w_1)\geq t_1 \wedge d(w_2)\leq t_2\right\} \qquad \textbf{(20)}$$

A first difference with the original algorithm is that the property $C_i \subseteq C_{i+1}$ is not required (and is violated by (**20**)) : $\Sigma(C_i)\geq LB(C_i)$ holds even if some edges of $C_i$ are not included in $C_{i+1}$ and inequalities on non-nested overlapping sets can coexist efficiently. In our case we sort intervals by increasing width in order to ensure a weaker property: $C_{i+1}\not\subset C_i$.

A second important point is that, when solving a sub-problem, remaining variables are not ignored but just continuously relaxed. Thus a global improvement constraint $\Sigma(A') < ub$ can be posted, where $ub$ is the value of the best solution found by a limited preliminary global search. This global cut prevents each interval to be optimized to the detriment of the remaining of the problem. Therefore the resolution of an interval is not only boosted by bounds on nested intervals but also by bound on other non-nested intervals, since improving the global polyhedron makes the global improvement cut more difficult to satisfy.

This whole procedure completes in less that 10 minutes on a 1GHz PC for instances described in Section 7. For instance on problem A1, once all intervals of widths 1 and 2 have been optimized, the corresponding cuts make the [6,9] interval infeasible: no assignation of binary values to $Z$ variables of [6,9] satisfies the improvement cut. This infeasibility proves the optimality of the solution found in the preliminary step. It shall be noted that removing valid inequalities of Section 4.3, only make this mechanism three times slower.

## 5. Ordering problem

In Section 3, the complexity of the FORMWORK PAIRING PROBLEM was proven using the one-family case and in Section 4 we solved a linear (aggregated) relaxation of the problem. In both cases ordering constraints are of no importance, therefore one could wonder whether this ordering aspect is a minor constraint or not, i.e. whether the pure ordering problem where formwork flow is fixed, is a difficult problem. In other words, once each shuttering is determined to be used on a fixed list of walls, is it difficult to arrange these shutterings on each wall in order to optimize the pairing?

If not, optimal solutions of *R*(FPP) (Section 4) may be polynomially extended to optimal solution of the FPP.

We prove in this section that no such polynomial algorithm exists unless P=NP (however section 6.2 presents a heuristic and efficient way of using solutions of the relaxation). Even the following local COMPATIBILITY problem, is NP-complete in the strong sense: for one wall, given the incoming and outgoing aggregated flows of formworks suggested by our relaxation, is it possible to arrange formworks in a sequence such that both incoming and outgoing subsets can be extracted from this sequence (as neighbors) ?

### 5.1. Definition

This COMPATIBILITY problem can be seen as a string problem, interpreting the set of formwork families as an alphabet of symbols. It aims at deciding whether there exists a string compatible with two decompositions (respectively representing incoming and outgoing formwork subsets of a wall).

NAME: COMPATIBILITY

INSTANCE: A finite alphabet $\Sigma = \{\varepsilon_1,\varepsilon_2, \dots \varepsilon_K\}$ and two collections of vectors of $\mathbf{N}^K$: $C=\{v_1,v_2, \dots v_n\}$ and $C'=\{v'_1,v'_2, \dots v'_m\}$.

QUESTION: Are $C$ and $C'$ "compatible", i.e. is there a string $z\in\Sigma^*$ and two permutations $\varphi$ and $\psi$ such that:

$z=z_1z_2\dots z_n$, concatenation of strings $z_i\in\Sigma^*$ $(i \le n)$, $z_i$ containing $v_{\varphi(i)}[k]$ times symbol $\varepsilon_k$ $\forall$ $k \le K$

$z=z'_1z'_2\dots z'_m$ concatenation of strings $z'_j\in\Sigma^*$ $(j \le m)$, $z'_j$ containing $v_{\psi(j)}[k]$ times symbol $\varepsilon_k$ $\forall$ $k \le K$

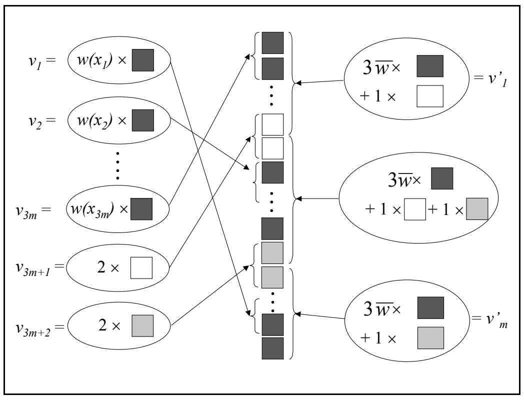
### 5.2. Complexity

**Proposition 3.** *COMPATIBILITY is NP-hard in the strong sense*

**Proof.** Consider an instance of 3-PARTITION (cf. 3.2) and transform it into the following COMPATIBILITY instance:

- $\Sigma = \{\varepsilon_1,\varepsilon_2\dots\varepsilon_m\}$
- $C = \{v_1,v_2,\dots v_{4m-1}\}$ with $\forall$ $i \le 3m$, $v_i=(w(x_i),0,\dots 0)$ and $\forall$ $i> 3m$, $v_i=(0,\dots 2,\dots 0)$ where the only non-null entry is $v_i[i-3m+1]=2$.
- $C'=\{v'_1,v'_2,\dots v'_m\}$ with: $v'_1=(3\overline{w},1,0,\dots 0)$, $v'_m=(3\overline{w},0,\dots 1)$ and $\forall j\in[2,m-1]$, $v'_j=(3\overline{w},0\dots,1,1,\dots 0)$, where the two consecutive "1" are at entries $\{j,j+1\}$.

The size of this instance is polynomial with the size of the original 3-PARTITION instance since it involves the same $w(x_i)$ numbers paired with $m$-1 zeros, $2m$-1 additional vectors (involving numbers 0,1 and $3\overline{w}$) and $m$ symbols.



$v_1 =$ $w(x_1) \times$
$v_2 =$ $w(x_2) \times$
$v_{3m} =$ $w(x_{3m}) \times$
$v_{3m+1} =$ $2 \times$
$v_{3m+2} =$ $2 \times$
$3\overline{w}\times$ $+1\times$ $= v'_1$
$3\overline{w}\times$ $+1\times$ $+1\times$
$3\overline{w}\times$ $+1\times$ $= v'_m$

**Fig. 4.** Transformation from 3-PARTITION to COMPATIBILITY

$\Rightarrow$: Assume C and C' are compatible. $\forall\ k\geq 2$, the compatibility string $z$ must contain $\varepsilon_k\varepsilon_k$ (vector $v_{3m+k-1}$). Since only $v'_{k-1}$ and $v'_k$ contain one $\varepsilon_k$, they must be neighbours in $z$ that is to say that $\psi$ must satisfy $|\psi^{-1}(k-1)-\psi^{-1}(k)|=1$. Thus $\psi$ is either the identity or "$\psi(j)=m+1-j$" and $z$ or its inverse equals $\varepsilon_1^{3\overline{w}}\varepsilon_2^2\varepsilon_1^{3\overline{w}}\varepsilon_3^2\varepsilon_1^{3\overline{w}}\varepsilon_4^2\ldots\ldots\varepsilon_1^{3\overline{w}}\varepsilon_m^2\varepsilon_1^{3\overline{w}}$. Each substring $\varepsilon_1^{3\overline{w}}$ covers substrings $z_{4k+1}z_{4k+2}z_{4k+3}$ of $z$ such that $v_{\varphi(4k+1)}[1]+v_{\varphi(4k+2)}[1]+v_{\varphi(4k+3)}[1]=3\overline{w}$. Hence $\{(x_{\varphi(4k+1)}, x_{\varphi(4k+2)}, x_{\varphi(4k+3)})\ |\ k\in[0,m]\}$ is a 3-partition of set $X$.

$\Leftarrow$: Assume reciprocally that there exists a collection $\{X'_1 \ldots X'_m\}$ of subsets of $X$, each of weight $3\overline{w}$ (and of cardinality 3) and define a permutation $\varphi$ on $[1,4\underline{m}-1]$ such that:

- $\forall\, j\in[1,m]$, $\forall\, i\in[4(j-1)+1, 4(j-1)+3]$ $x_{\varphi(i)}\in X'_j$
- $\forall\, j\in[1,m-1]$ $\varphi(4j) = 3m+j$

String $z=\varepsilon_1^{3\overline{w}}\varepsilon_2^2\varepsilon_1^{3\overline{w}}\varepsilon_3^2\varepsilon_1^{3\overline{w}}\varepsilon_4^2\ldots\ldots\varepsilon_1^{3\overline{w}}\varepsilon_m^2\varepsilon_1^{3\overline{w}}$, with permutations $\varphi$ and $\psi$=*identity* proves the compatibility of C and C'.

Finally 3-PARTITION can be polynomially transformed into COMPATIBILITY, thus COMPATIBILITY is NP-hard in the strong sense. ■

A polynomial specific case of COMPATIBILITY is given in appendix as well as an NP membership proof (since the size of compatibility string $z$ is not polynomial with the instance size, a more compact certificate of compatibility must be exhibited). We refer the reader to [2] for a more detailed study of this problem.

Finally, trying to extend solutions of $R$(FPP) to solutions of the FPP would require solving $N$ NP-hard COMPATIBILITY problems subject to coupling constraints (for each edge $w_1\rightarrow w_2$, sequences chosen for walls $w_1$ and $w_2$ must induce identical orders on formworks of this edge). This difficulty advocates for the design of heuristic approaches to the FPP (Section 6), possibly using solutions of $R$(FPP) as an oracle.

## 6. Greedy Algorithm and Tabu Search for the FPP

A naive greedy algorithm consists in arbitrary ordering formworks on first day walls, then using the biggest possible formwork chain from one of these walls for a second day wall, then the second biggest, etc… Applying such a strategy ($Greedy_{FPP}$) on a construction site means re-using each day the biggest possible formwork assemblies and randomly dispatching the remaining (without studying what ordering would be preferable for next days): it approximately divides the number of junctions by two, compared to moving shutterings one by one ($1by1_{FPP}$).

This section describes a Tabu approach inspired from [9]. It is based on a compact model focused on parallel edges. These pairings are inserted one by one into a pairing set (Section 6.1), ejecting conflicting ones if any (Section 6.3). A Tabu algorithm controls this insertion/ejection heuristic (Section 6.2).

### 6.1. Pairing model

We define a pairing $\gamma$ as a triplet of $P\times[1,K]\times[1,K]$ (cf. equation (**3**)). A pairing $\gamma=\{(w_1,p_1)\rightarrow(w_2,p_2),k,k'\}$ models a triplet of constraints $\{Y_{(w1,p1)\rightarrow(w2,p2)}=1$, $X^{(k)}_{(w1,p1)\rightarrow(w2,p2)}=1$, $X^{(k')}_{(w1,p1+1)\rightarrow(w2,p2+1)}=1\}$. Such a pairing represents a pair of formworks moved together. It *constrains* nodes $(w_1,p_1)$ and $(w_2,p_2)$ to be crossed by commodity $k$, and nodes $(w_1,p_1+1)$ and $(w_2,p_2+1)$ to be crossed by commodity $k'$. It is said to *overlap* another pairing γ' if $w_1=w'_1$, $w_2=w'_2$, $p_1+1=p'_1$ and $p_2+1=p'_2$ that is to say that γ,γ' form a triplet of formworks moved together. Finally we note $E_\gamma=\{(w_1,p_1)\rightarrow(w_2,p_2),\ (w_1,p_1+1)\rightarrow(w_2,p_2+1)\}$. A pairings set Γ is *consistent* if and only if it satisfies the following four constraints:

i. All pairings constraining the same node $(w,p)$ must constrain it to be crossed by the *same* commodity, because only one formwork can be the $p^{th}$ on wall $w$.
ii. Non-overlapping pairs $(\gamma, \gamma') \in \Gamma^2$ must satisfy: $w_i\neq w'_i \vee |p_i-p'_i|\geq 2,\ \forall\ i\in\{1,2\}$. Indeed if γ and γ' share a formwork at their origin for instance, they must move this shared item to the *same* position i.e. they must overlap.
iii. For all wall $w$, $\forall\ k \leq K$, the number of positions constrained to be crossed by commodity $k$ must be smaller than $a_w[k]$, otherwise the allocated number of formworks of family $k$ would be exceeded on wall $w$.
iv. For each day and each formwork family, the allocations $a_w[k]$ of this day, plus the number of formworks of type $k$ waiting on previous walls to be re-used on next days cannot exceed available quantities $A[k]$:

$$\forall t \leq T, \forall k \leq K, \sum_{w\in W_t} a_w[k] + \sum_{\left\{(w_1,p_1)\rightarrow(w_2,p_2)\in\bigcup_{\gamma\in\Gamma} E_\gamma \middle|\ d(w_1)<t\wedge d(w_2)>t\right\}} X^{(k)}_{(w_1,p_1)\rightarrow(w_2,p_2)} \leq A[k] \qquad \textbf{(21)}$$

In terms of the FPP multi-flow, constraint iv (**21**) makes sure that the flow crossing each layer (either through nodes or arcs) never exceeds capacities $A[k]$, $k\leq K$.

**Proposition 4.** *There is a* surjection *from the set of FPP solutions to the set of all* consistent *sets of pairings, mapping each FPP solution of cost c to a consistent set of size $\Sigma|a_w|$-N-c.*

**Proof.** From any FPP solution, reading the corresponding pairing set is straightforward; its size is exactly the number of saved junctions $\Sigma y$. And for any consistent pairing set of size $s$, a flow satisfying the $3s$ constraints modeled by these $s$ triplets is easy to find, assigning to each unconstrained node-position (from left to right) available commodities from previous nodes, equation (**21**) avoiding starvation. The cost of this FPP solution is at most $\Sigma|a_w|\text{-}N\text{-}s$. ■

The advantage of this pairing model is to focus on the important part of the problem (parallel arcs), breaking a lot of symetries. Besides, in a local search approach, it means that any consistent set is a feasible solution that can be locally modified adding or removing pairings.

## 6.2. Insertion/Ejection Tabu search

In [9] radio links are inserted in the frequency range and interfering links are ejected and added to the Tabu list. Similarly we start with $\Gamma=\varnothing$ and insert pairings one by one, conflicting ones being ejected and added to the Tabu list. More precisely, each move consists in selecting a (non-tabu) pairing $\gamma$ to insert, and removing an ejection set $S_\gamma \subseteq \Gamma$ from $\Gamma\cup\{\gamma\}$ such that $\Gamma\cup\{\gamma\}\backslash S_\gamma$ is consistent. The computation of a (nearly) minimum-cardinality ejection set is detailed in section 6.3.

We compared two version of this Tabu search. The first one (denoted $Tabu_{FPP}$) uses an heuristic inspired from the greedy algorithm to add non-conflicting parings. Once no more pairing can be added without conflict (local optimum), we randomly generate a bunch of pairings and insert the one whose associated ejection set has minimum size. The second approach (denoted $GuidedTabu_{FPP}$) is a variant using $R$(FPP) as an oracle: pairings are inserted accordingly to the optimal solution of this multiflow relaxation. More precisely, if $X^{(k)}_{w1\rightarrow w2}$ items of type $k$ are transported from wall $w_1$ to $w_2$ in the $R$(FPP) solution, this number is taken as an upper bound in the tabu search: any candidate pairing whose insertion would make this bound exceeded is considered as conflicting. The results of this algorithms are discussed in section 7.

## 6.3. Ejection Set computation

Adding a pairing $\gamma$ to a consistent set $\Gamma$ may lead to an inconsistent set. We consider here the problem of computing a minimum-cardinality ejection set $S_\gamma\subseteq\Gamma$ such that $\Gamma\cup\{\gamma\}\backslash S_\gamma$ is consistent. Such a set exists for any pairing $\gamma$ such that $\{\gamma\}$ is consistent, since $\Gamma$ is always a valid ejection set.

Physical constraints i and ii require to eject the corresponding conflicting pairings. On the contrary there are several ways to solve conflicts associated to cumulative constraints iii and iv (equation (**21**)). Any pairing $\gamma$ causes at most 4 *covering* conflicts (constraint iii) and $2\times(start(w_2)\text{-}end(w_1))$ *availability* conflicts (constraint iv). A *covering* conflict occurs on a wall $w$ when too many positions are constrained to be covered by a formwork of family $k$: thus one or two positions must be freed, removing the (at most 4) pairings constraining it. Similarly an *availability* conflict occurs for a day $t$ and a family $k$ when too many formworks of this family are planned to be moved from a previous wall to a future one (thus unavailable on day $t$): one or two of these shutterings must be made available on day $t$, removing the (at most 2) corresponding pairings. We solve this problem with a (sub optimal) greedy algorithm removing the pairing subset solving the maximum number of conflicts (lookahead), and repeating this step until all conflicts are solved.

## 7. Computational results

The following table lists the results obtained on 16 FPP instances (available at http://e-lab.bouygues.com) where A0 is the 10-wall example of **Fig. 2** and others are real instances of *Bouygues Habitat*, involving an average of 100 walls. The three rightmost columns give the three lower bounds computed on the relaxation *R*(FPP) of section 4: the continuous optimum of the LP model (with and without *Xpress* automatically generated cuts) and its optimal integer solution obtained by our *RDS* strategy (in less than 10 minutes as pointed out in Section 4.4). The left columns correspond to the four resolution algorithms tested: $1by1_{FPP}$, $Greedy_{FPP}$, $Tabu_{FPP}$ and $GuidedTabu_{FPP}$. Less that 1 second CPU is required by $1by1_{FPP}$ and $Greedy_{FPP}$. As for the local search approaches, both were stopped after 10000 moves i.e. around 5 minutes.

Table 1. Computational results

| **Data** | | | | **Upper Bounds (solutions)** | | | | **Lower Bounds** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Problem* | *Days* | *Walls* | *Families* | $1by1_{FPP}$ | $Greedy_{FPP}$ | $Tabu_{FPP}$ | *Guided* $Tabu_{FPP}$ | *RDS* | *LP with Cuts* | *LP* |
| A0 | 5 | 10 | 6 | 25 | 17 | 14 | **14** | **14** | 14 | 14 |
| A1 | 10 | 76 | 7 | 174 | 91 | 78 | 76 | 74 | 72 | 69 |
| B0 | 10 | 76 | 21 | 209 | 114 | 94 | 91 | 87 | 85 | 81 |
| B3 | 8 | 80 | 9 | 145 | 86 | 76 | 75 | 73 | 72 | 70 |
| B4 | 6 | 27 | 9 | 48 | 35 | 31 | **31** | **31** | 31 | 31 |
| B5 | 8 | 79 | 9 | 128 | 72 | 66 | **66** | **66** | 65 | 64 |
| B6 | 26 | 154 | 9 | 123 | 73 | 61 | **60** | **60** | 60 | 58 |
| B9 | 26 | 125 | 9 | 60 | 37 | 33 | **33** | **33** | 33 | 31 |
| C1 | 11 | 69 | 9 | 124 | 69 | 63 | **63** | **63** | 63 | 61 |
| C2 | 19 | 143 | 18 | 152 | 68 | 63 | 61 | 56 | 55 | 52 |
| C3 | 8 | 80 | 9 | 151 | 84 | 73 | 73 | 68 | 67 | 65 |
| C4 | 7 | 45 | 21 | 45 | 33 | 28 | **28** | **28** | 28 | 28 |
| C5 | 8 | 79 | 9 | 131 | 74 | 67 | 67 | 66 | 65 | 64 |
| C6 | 25 | 153 | 8 | 259 | 129 | 120 | 116 | *111*[6] | 110 | 106 |
| C7 | 19 | 143 | 18 | 171 | 80 | 64 | 64 | 59 | 59 | 56 |
| C9 | 26 | 125 | 9 | 89 | 50 | 42 | **41** | **41** | 41 | 39 |

These results prove that the natural greedy approach to the (NP-hard) FORMWORK PAIRING problem ($Greedy_{FPP}$) is far from being optimal: the proposed $Tabu_{FPP}$ based on a compact model of the problem produces significantly better solutions. Besides, solving a multiflow relaxation of the problem with a Russian-Dolls strategy provides lower bounds that establish the optimality of the $Tabu_{FPP}$ solution on 6 of the 16 instances. Finally, when using an optimal solution of this relaxation as a guide to our tabu search ($GuidedTabu_{FPP}$), two other instances are optimally solved and 5 solutions are improved. In summary 8 instances remain open with gaps ranking from 1.5% to 9%.

[6] On C6, our RDS scheme was not sufficient to reach the optimum of the relaxation: 111 is merely a lower bound.

## 8. Conclusion

In this paper we have introduced the FORMWORK PAIRING PROBLEM, arising in construction sites. We have proven its strong NP-hardness and designed a Tabu algorithm producing good solutions. Solving an adequate relaxation with an innovative "Russian Dolls" approach, we obtained both a lower bound and an oracle. This oracle proved to be useful as a guide for the Tabu algorithm, and the lower bound proved the optimality of 50% of our solutions.

Compared to the greedy approach, our best Tabu algorithm saves an average of 14% of the number of junctions to be performed. From an industrial point of view it means that the corresponding labor cost can be decreased by 14% through formwork pairing optimization.

## Acknowledgement

The author is grateful to Fabrice Chauvet for his precious help on complexity proofs.

## References

[1] J. Agnèse, N. Bataille, E. Bensana, D. Blumstein and G. Verfaillie. *Exact and Approximate methods for the Daily Management of an Earth Observation Satellite*. In Proc. of the 5th ESA Workshop on Artificial Intelligence and Knowledge Based Systems for Space 1995.
[2] T. Benoist and F. Chauvet: *Complexity of some FPP related Problems*. E-lab Technical Report, 2001.
[3] Y. Caseau and F. Laburthe. *Improved CLP Scheduling with Task Intervals*. In Proc. of the 11th International Conference on Logic Programming, MIT Press, 1994
[4] M.R. Garey and D.S. Johnson: *Complexity results for multiprocessor scheduling under resource constraints*. SIAM J. Comput. 4, 397-411, 1975.
[5] M.R. Garey and D.S. Johnson: *Computers and intractability, a guide to the theory of NP-completeness*. New York: W. H. Freeman, 1979.
[6] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. *Lifted flow cover inequalities for mixed 0-1 integer programs*. Vol. 85/3, pp 439-467 Mathematical Programming, 1999.
[7] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988. Wiley Interscience Series in Discrete Mathematics and Optimization.
[8] G. Verfaillie, M. Lemaître, and T. Schiex. *Russian Doll Search for Solving Constraint Optimization Problems*. In Proc. of AAAI-96, pages 181--187, Portland, OR, 1996.
[9] M. Vasquez. *Challenge Roadef 2001*. In Francoro III, Quebec 2001, to appear in EJOR.

## Appendix: Properties of the COMPATIBILITY Problem

When there is at most one occurrence of each symbol in the researched string $z$, one can reformulate the question as follows: naming $X$ the set of symbols appearing in $z$, is it possible to order symbols of $X$ such that each element (subsets of $X$) of $C \cup C'$ appears in a consecutive block?

Let $G$ be a bipartite graph defined as follow: left vertices represent elements (subsets of $X$) of collection $C$, right vertices represent elements of collection $C'$, and for each pair $(c,c') \in C \times C'$ the corresponding edge is added in $G$ if and only if these subsets intersect without one being included in the other i.e. $(c \cap c' \neq \emptyset) \wedge (c \not\subset c') \wedge (c' \not\subset c)$

**Proposition 5.** *C and C' are compatible if and only if G contains no node of degree greater or equal to 3 and no cycle.*

**Proof.** See [2] (arcs in $G$ represent *neighbours* in the compatibility string $z$, hence only degrees 1 and 2 are allowed and cycles are forbidden).

Finally (in this "all different" case) the compatibility question can be decided computing *0(mn)* intersections of subsets ($c \cap c'$).

This polynomial case of COMPATIBILITY can be used to prove its membership to NP. Given *C* and *C'*, two compatible vectors, let *M* be a 3-dimensional matrix such that $\forall\ i \le n$, $\forall\ j \le m$, $\forall\ k \le K$, *M[i,j,k]* is the number of $\varepsilon_k$ shared by $v_i$ and $v'_j$ on string *z*. Such a matrix describes the dispatching of "symbols sets" of *C* into "symbol sets" of *C'* and is valid if and only if *M* abides:

$$\forall i \le n, \sum_{j \le m} M[i,j] = v_i \quad \text{and} \quad \forall j \le m, \sum_{i \le n} M[i,j] = v'_j \tag{22}$$

**Proposition 6.** *Matrix M is a polynomial size compatibility certificate whose validity can be checked in polynomial time.*

**Proof.** See [2] (*M* proves the compatibility of *C* and *C'* when the bipartite graph linking vertices $i \le n$ and $j \le m$ such that $(M[i,j] \neq 0) \wedge (M[i,j] \neq v_i) \wedge (M[i,j] \neq v_j)$ contains no cycle and no node of degree $\geq 3$).

Finally, we conclude that COMPATIBILITY belongs to NP.