Complexity of some FPP related Problems

Thierry Benoist[1], Fabrice Chauvet[2]

[1]Bouygues e-lab, 1 av. Eugène Freyssinet,
78061 St Quentin en Yvelines Cedex, France
tbenoist@bouygues.com
[2]Bouygues Telecom R&D, 30 avenue de l'Europe,
F78944 Vélizy Cedex, France
fchauvet@bouyguestelecom.fr

e-lab Research Report, December 2001

Abstract. This report establishes the NP-completeness of two problems issued from the FORMWORK PAIRING PROBLEM (FPP). The first one is the bipartite case of the MINIMUM EDGE -COST FLOW problem (which general case what proved NP-complete by Garey & Johnson). The second is a string problem consisting in deciding whether there exists a string compatible with two given decompositions.

1. Introduction

This report establishes the NP-completeness of two problems issued from the FORMWORK PAIRING PROBLEM (FPP)[B]. The first one is the bipartite case of the MINIMUM EDGE-COST FLOW problem [GJ79]. The second is a string problem consisting in deciding whether there exists a string compatible with two given decompositions. We named this second problem COMPATIBILITY.

Both problems are fully defined in this report but their relevancy stems from their relation to the FPP (briefly described in section 2). The BIPARTITE MINIMUM EDGE-COST FLOW models a two-days FPP with only one formwork family. The COMPATIBILITY problem formalizes the following question: given the sets of formworks one wants to use to frame a wall (incoming sets) and the sets of formwork one wants to extract for later reuse (outgoing sets), is it possible to concatenate incoming sets in such an order that outgoing sets can be extracted?

2. The FORMWORK PAIRING PROBLEM

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 within about ten days, using a limited amount of vertical formworks whose rotation and assembling are planned by the Method Department. 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 the shuttering can be removed and reused elsewhere on the site. Different formwork models with several lengths are used on a construction site but walls are usually longer than the maximal length: therefore covering a wall requires *assembling* several models together. Given the construction schedule i.e. the lists of walls to be framed on each day, computing the minimum set of formworks to order for the site (under industrial constraints) is an optimization problem per se that is not addressed in this paper. Thus we consider here that this preliminary problem has been solved and that the set of formworks assigned to each wall is known.

Assembling two shutterings together takes several minutes and consumes 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), determining the successive positions (wall + position on the wall) of each formwork of each family in order to minimize the sum of costs defined on each wall by the number of formwork pairs of the wall assembly that were not already neighbours on their previous positions. This cost is exactly the number of junctions to be performed.

A more comprehensive description of the FPP can be found in [B].

3. BIPARTITE MINIMUM EDGE-COST FLOW

The general case of the MINIMUM EDGE-COST FLOW was proved to be NP-complete in [GJ79]. We recall this proof in section 3.1 in order to show that it does not address the complexity of the bipartite case. The NP-completeness of this variant is then proved in sections 3.2 and 3.3 (strong sense).

3.1. Definition and Complexity of the general case

The MINIMUM EDGE-COST FLOW is the following.

NAME: MINIMUM EDGE-COST FLOW

INSTANCE: A directed graph G = (V,A) a supply/demand function $s: V \mapsto Z^+$ and an integer B.

QUESTION: Is there a flow function $f : A \mapsto Z^+$ such that:

$$\forall x \in V, \sum_{(x,y)\in A} f(x,y) - \sum_{(y,x)\in A} f(y,x) = s(x) \quad (1)$$

$$\text{and } |\{(x,y)\in A \mid f(x,y)>0\}| \leq B \quad ? \quad (2)$$

[GJ79] suggest proving the complexity of MINIMUM EDGE-COST FLOW by a transformation from X3C (EXACT COVER BY 3-SETS). This can be done as follow.

Theorem 1. *MINIMUM EDGE-COST FLOW is NP-Complete in the strong sense.*

Proof. An instance of X3C is composed of a finite set X of *3m* elements and a collection C of 3-elements subsets of X. Deciding whether C contains an exact cover for X (a subcollection C such that each $x \in X$ occurs in exactly one member of C) is NP-complete in the strong sense [K72].

Any instance of X3C can be polynomially transformed in an instance of MINIMUM EDGE-COST FLOW by defining one sink (denoted v_i) per element of X (destination of 1 unit of flow), one node (denoted u_i) per 3-set of C (linked to the corresponding 3 sinks by edges of capacity 1), and one source (denoted *s*) of *3m* units of flow linked to all 3-sets' nodes by edges of capacity 3. The goal is then to find a flow activating less than *4m* edges.

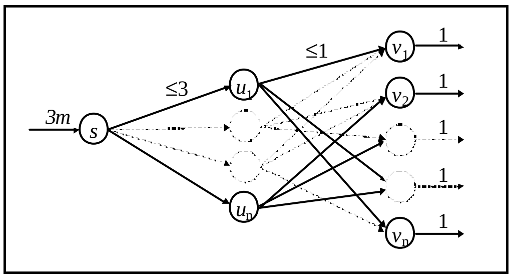

Fig. 1. Transformation from X3C

Þ : If C'⊆ C is an exact cover for X, then the flow using only edges from the source to nodes corresponding to 3-sets of C' is feasible (since each sink receive exactly one unit of flow) and activates *4m* edges (*m* from the source to "active" nodes, and one to each of the *3m* sinks).

Ü : Reciprocally if *F* is a flow activating less than *4m* edges, then it activates exactly *m* edges from the source since each sink requires at least one incoming edge and transporting *3m* units of flow from the source through 3-capacited edges required at least *m* edges. What remains to prove is that the 3-sets corresponding to this *m* "active" nodes are an exact cover of C. Each x ∈X appears at least in one of this 3-sets (receive its unit of flow). Besides no x ∈X appears in more than one 3-set because each "active" node receive 3 units of flow, and must send one to each of its elements (hence any x ∈X appearing in several 3-sets would receive several units of flow, in contradiction with its unitary sink capacity).

Finally MINIMUM EDGE-COST FLOW is NP-complete in the strong sense. ■

3.2. Definition and Complexity of the bipartite case

The BIPARTITE MINIMUM EDGE-COST FLOW is a special case of the MINIMUM EDGE-COST FLOW where the graph is bipartite. It consists in finding a transportation solution using at most B arcs. It models a two-days FPP with only one formwork family.

It shall be noted that the reduction of section 3.1 is not operational for this special case 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 is impossible when flows across all nodes are fixed.

NAME: BIPARTITE MINIMUM EDGE-COST FLOW

INSTANCE: A bipartite graph G = (U∪V, A), a function b:U∪V → Z^+ and a positive integer B.

QUESTION: Is there a flow function $f : A \mapsto Z^+$ such that:

$$\forall x \in U \sum_{y \in V} f(x,y) = b(x),\ \forall y \in V \sum_{x \in U} f(x,y) = b(y),\ |\{(x,y) \in A \mid f(x,y) > 0\}| \leq B \qquad (3)$$

We will prove the NP-Completeness of this problem by a polynomial transformation from SUBSET SUM.

Theorem 2. *BIPARTITE MINIMUM EDGE-COST FLOW is NP-Complete.*

Proof. An instance of SUBSET SUM is composed of a finite set X of n positively weighted elements (x_i weighting w_i) and a positive integer B. The question whether there is a subset X' ⊆ X, such that the sum of weights of elements of X' is exactly B is NP-complete in the ordinary sense [K72].

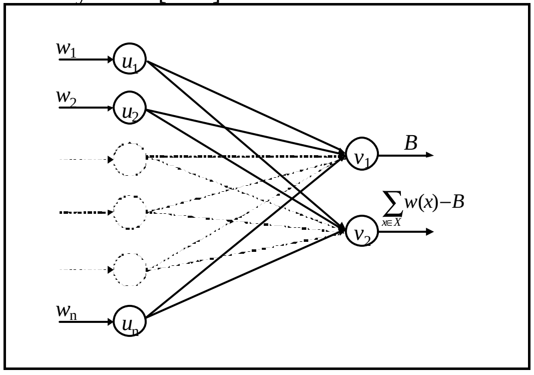

Fig. 2. Transformation from SUBSET SUM

To each element x_i of X we associate a left vertex u_i, source of a flow equal to w_i. Two right vertices are created: v_1 is a sink of a flow equal to B, v_2 is a sink of a flow equal to $B - \sum_{x \in X} w(x)$. We consider the complete bipartite graph G with infinite capacities on all edges, and the problem of finding a flow in this graph with less than n active edges.

Þ **:**Assume X has a subset X’ whose sum equals B. Then if we name U’ the corresponding subset of left vertices, the flow using only edges from U’ to v_1 and from U \ U’ to v_2 is trivially feasible and only activates n edges.

Ü **:**Assume reciprocally that F is a flow on G activating only n edges. Since no weight is null, we can infer that for each left vertex u there is exactly one active edge (u,v_1) or (u,v_2). If we name U’ the set of left vertices u' such that (u',v_1) is active, then the corresponding subset X’ $\subseteq$ X has a weight equal to B.

Finally BIPARTITE MINIMUM EDGE-COST FLOW is NP-complete. ■

3.3. Strong sense

The above transformation can be generalized to a transformation from 3-PARTITION in order to prove that the problem is NP-Complete *in the strong sense*

Theorem 3. *BIPARTITE MINIMUM EDGE-COST FLOW is NP-Complete in the strong sense.*

Proof. Let us consider the NP-complete 3-PARTITION problem [GJ75]:

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

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

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 (**4**) enforces these subsets to contain exactly three elements).

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

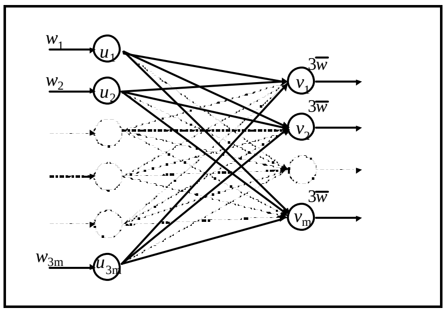

Fig. 3. Transformation from 3-PARTITION

Each $x_i \in$ X is associated to a left vertex u_i source of a flow w_i. We create m right vertices v_1 .. v_m,each of which is a sink of a flow $3\overline{w}$, 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.

Þ : Assume $X_1, X_2, \ldots X_m$ is a 3-partition of X (each X_j 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 from U_j to v_j for all *j* is trivially feasible and only activates *3m* edges.

Ü : 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_j). If for each *j* we name U_j the set of left vertices *u* such that (u,v_j) is active, then the corresponding subsets $X_1, X_2, \ldots X_m$ are a 3-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 smaller than three. ■

A corollary of theorem 3 is that MINIMUM EDGE-COST FLOW remains NP-complete in the strong sense when the flow across each node is fixed.

4. COMPATIBILITY

The COMPATIBILITY problem is a string problem, aiming at deciding whether there exists a string compatible with two decompositions (respectively representing "incoming" and "outgoing" symbol groups). It occurs on construction sites when, knowing the collection of (non-ordered) sets of formworks that we intend to assemble to frame a wall, we wonder whether these formworks can be ordered in such a way that a certain collection of sets will be extractable from the wall for future reuse.

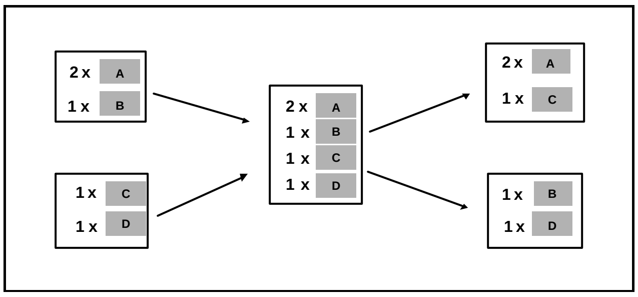

Fig. 4. A non-compatible case

Fig. 4 shows an example where *no* ordering of the five formworks of types A (twice), B, C and D can both:

- result of the junction of sets "2×A+B" and "C+D",
- and be partitioned into the two sets: "2×A+C" and "B+D".

In other words, collections {"2×A+B","C+D} and {"2×A+C" and "B+D"} are not compatible.

On the contrary, collections {"2×A+B","C+D} and {"2×A" and "C+B+D"} are compatible: for instance the order AABCB both equals AAB-CD and AA-BCD.

We formally define this problem in section 4.1 and propose in section 4.2 a polynomial algorithm dedicated to the case where all symbols of the researched string are different. We use this result to establish in section 4.3 that COMPATIBILITY belongs to NP, and prove its NP-completeness in sections 4.4 and 4.5 (strong sense). Finally section 4.6 proves the strong NP-completeness of the particular case of a two-symbols alphabet.

4.1. Definition

NAME: COMPATIBILITY

INSTANCE: A finite alphabet $\Sigma = \{\varepsilon_1, \varepsilon_2, \dots \varepsilon_K\}$ and two collections of vectors of 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 φ and ψ such that:

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

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

4.2. "All different" case

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 of C∪C' (subsets of X) 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.

$$(c \cap c' \neq \emptyset) \wedge \neg (c \subseteq c') \wedge \neg (c' \subseteq c) \qquad \textbf{(5)}$$

Lemma 1. C and C' are compatible if and only if G contains no node of degree ≥ 3 and no cycle.

Proof.

Þ **:** Assume C and C' are compatible. The existence of edge (c,c') requires c and c' to be *neighbours* in the ordering, i.e. $(c \cap c')$ appears between (c-c') and (c'-c), none of these 3 sets being empty. Therefore no subset (element of C∪C') can have more than two neighbours (what forbids nodes with degree 3 or greater) and the existence of cycles is forbidden.

Ü **:** Reciprocally, if these conditions are satisfied, edges form a set of paths. We will prove that each of these paths trivially leads to a valid ordering. For any subset *s* of X we note <*s*> an arbitrary ordering of elements of *s*, and $\forall$ *c*∈C we note *sub(c)* the string obtained by an arbitrary concatenation of sub-strings <c'> of all sets of C' fully included in *c*. With these notation, any sub-path c'_1-c-c'_2 can be replaced by $<c'_1 \cap c>$-sub(c)-$<c \cap c'_2>$.Finally the concatenation of these sub-ordering builds a valid ordering proving the compatibility of C and C'.

Since the computation of paths (possibly detecting inconsistencies) can be done simultaneously to the creation of graph G, the compatibility question can be decided computing *0(mn)* intersections of subsets ($c \cap c'$). ■

4.3. Membership to NP

Membership to NP does not stem straightforward from definition stated in 4.1 since the size of compatibility string *z* is not polynomial with the instance size. Thus a more compact certificate of compatibility must be exhibited.

Theorem 4. *COMPATIBILITY Î NP*

Proof. Given C and C', two compatible vectors, let $M \in Z^3$ be a 3-dimensional matrix such that $\forall\ i \leq n$, $\forall\ j \leq m$, $\forall\ k \leq K$, *M[i,j,k]* is the number of ε_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:

$$\forall i \leq n, \sum_{j \leq m} M[i,j] = v_i \quad \text{and} \quad \forall j \leq m, \sum_{i \leq n} M[i,j] = v'_j \qquad (\mathbf{6})$$

Matrix *M* allows distinguishing identical symbols by their origin in C and their destination in C', leading to an equivalent alphabet $\Sigma^+ = \{\varepsilon_{ijk} \mid M[i,j,k] \neq 0\}$. Finally identical symbols all having same origin and destination can be arbitrary distinguished with an additional *l* index. With the resulting Σ^{++} alphabet, each symbol ε_{ijkl} occurs exactly once in C and C', and the compatibility of C and C' can be checked in polynomial time with the algorithm described in 4.2 (Lemma 2).

In other words, matrix M is a polynomial size compatibility certificate if and only if the bipartite graph with *n* left vertices and *m* right vertices, linking vertices *i* and *j* when:

$$M[i,j] \neq 0_K,\ M[i,j] \neq v_i \text{ and } M[i,j] \neq v_j \quad (\text{with } 0_K[k]=0\ \forall\ k \leq K) \qquad (\mathbf{7})$$

contains no cycle and no node of degree ≥ 3.

Since equations (**6**) and (**7**) can be checked in polynomial time, we conclude that COMPATIBILITY belongs to NP. ■

4.4. NP-completeness

Theorem 5. *COMPATIBILITY is NP-Complete*

Proof. Consider an instance of the SUBSET SUM problem (set $X = \{x_1, x_2, \ldots x_n\}$, weights $\{w_1, w_2, \ldots w_n\}$, total weight W, positive integer B) and transform it into the following COMPATIBILITY instance:

- $\Sigma = \{\varepsilon_1, \varepsilon_2\}$
- $C = \{(w_i, 0) \mid i \in [1,n]\} \cup \{(0,2)\}$
- $C' = \{(B,1), (W-B,1)\}$

The size of this instance is polynomial (linear) with the size of the original SUBSET SUM instance since it involves the same w_i numbers paired with 0, 3 additional pairs (involving numbers B and W) and 2 symbols.

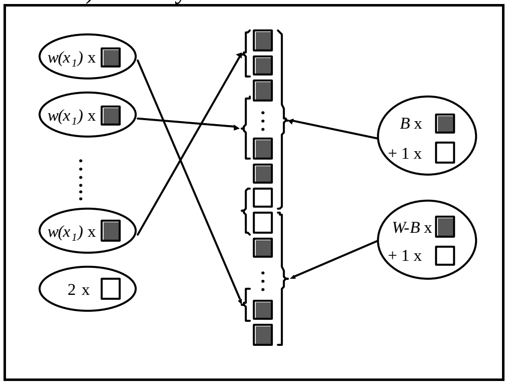

Fig. 5. Transformation from SUBSET SUM to COMPATIBILITY

Þ **:** Assume C and C' are compatible, with compatibility string $z \in \Sigma^*$ and permutations φ and ψ, and consider without loss of generality (for symmetry reasons) that ψ is the identity. With $\tilde{n} = \varphi^{-1}(n+1)$ (such that $z_{\tilde{n}} = \varepsilon_2\varepsilon_2$), $z'_1 = z_1 z_2 \ldots z_{\tilde{n}-1}\varepsilon_2$. Therefore, $X' = \{x_{\varphi(i)} \mid i \in [1, \tilde{n}-1]\}$ is a subset of X whose weight is exactly B.

Ü **:** Assume reciprocally that there exists a subset X' of weight B, note its cardinality ñ and define a permutation φ on [1,n+1] such that:

- $\forall\, i < \tilde{n}\ \ x_{\varphi(i)} \in X'$
- $\varphi(\tilde{n}) = n+1$
- $\forall\, i > \tilde{n}\ \ x_{\varphi(i)} \notin X'$

String $z = \varepsilon_1^{B}\varepsilon_2^{2}\varepsilon_1^{W-B}$, with permutations φ and *identity* proves the compatibility of C and C'.

Finally SUBSET SUM can be polynomially transformed into COMPATIBILITY, thus COMPATIBILITY is NP-Complete, even with K=2, and one of the symbols limited to 2 occurrences. ■

4.5. Strong sense

Theorem 6. *COMPATIBILITY is NP-complete in the strong sense*

Proof. Consider an instance of 3-PARTITION (cf. section 3.3) and transform it into the following COMPATIBILITY instance:

- $\Sigma = \{\varepsilon_1, \varepsilon_2 \ldots \varepsilon_m\}$
- $C = \{v_1, v_2, \ldots v_{4m-1}\}$ with $\forall\, i \leq 3m$, $v_i = (w_i, 0, \ldots 0)$ and $\forall\, i > 3m$, $v_i = (0, \ldots 2, \ldots 0)$ where the only non-null entry is $v_i[i-3m+1] = 2$.
- $C' = \{v'_1, v'_2, \ldots v'_m\}$ with: $v'_1 = (3\overline{w}, 1, 0, \ldots 0)$, $v'_m = (3\overline{w}, 0, \ldots, 0, 1)$ and $\forall j$ *Î [2,m-1]*, $v'_j = (3\overline{w}, 0 \ldots, 1, 1, \ldots. 0)$, where the two consecutive ones 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_i numbers paired with m-1 zeros, 2m-1 additional vectors (involving numbers 0,1 and $3\overline{w}$) and *m* symbols.

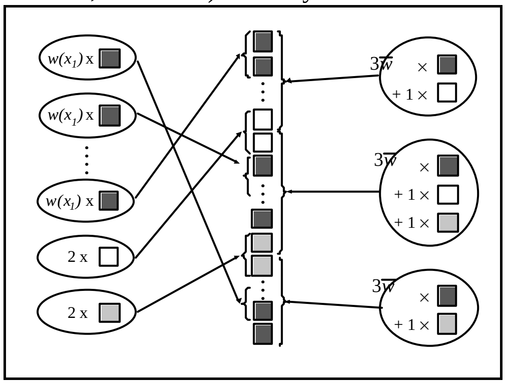

Fig. 6. Transformation from 3-PARTITION to COMPATIBILITY

Þ : Assume C and C' are compatible. The extraction of v_i, $i>3m$ requires an order such that each v'_j has neighbours v'_{j-1} and v'_{j+1}. Therefore ε_1 symbols of each v'_j are separated from others by pairs of other symbols. Thus each v_i, $i\leq 3m$ is assigned to a unique v'_j. This assignation induces a 3-partition of the corresponding weighted set X.

Ü : Assume reciprocally that there exists a collection $\{X'_1 \dots X'_m\}$ of subsets of X, each of weight $3\overline{w}$ (and of cardinality 3) and define a permutation φ on [1,4m-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 $z = e_1^{3\overline{w}} e_2^{2} e_1^{3\overline{w}} e_3^{2} e_1^{3\overline{w}} e_4^{2} \dots\dots e_1^{3\overline{w}} e_m^{2} e_1^{3\overline{w}}$, with permutations φ and *identity* proves the compatibility of C and C'.

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

4.6. Strong NP-completeness of the two symbols case

We prove in this section that COMPATIBILITY remains NP-complete in the strong sense when the number of symbols is limited to two.

3-PARTITION can be transformed into:

- $\Sigma = \{\varepsilon_1, \varepsilon_2\}$
- $C = \{v_1, v_2, \dots v_{4m-1}\}$ with $\forall\, j \leq 3m$, $v_j = (w(x_j), 0)$ and $\forall\, j > 3m$, $v_j = (0, 2^{2(j-3m)-1})$
- $C' = \{v'_1, v'_2, \dots v'_m\}$ with: $v'_1 = (3\overline{w}, 1)$, $v'_m = (3\overline{w}, 2^{2m-4})$ and $v'_i[i] = (3\overline{w}, 2^{2i-4} + 2^{2i-2})1\ \forall\, i \in [2,m-1]$.

This instance size is polynomial with that of the 3-PARTITION instance since entries of vectors of $C \cup C'$ are bounded by 2^{2m-3} that can be coded in less than *2m* bits.

Assume C and C' are compatible.

Lemma 2. *The only way to extract v_{4m-1} $(0,2^{2m-3})$ is to make v'_{m-1} and v'_{m} neighbours.*

Proof. v'_{m-2} and v'_{m-1} do not contain enough ε_2 to form v_{4m-1} :

$$v'_{m-2}[2] + v'_{m-1}[2] = 2^{2m-8}+2\times2^{2m-6}+2^{2m-4} = (2^{-5}+2^{-2+}+2^{-1})\, v_{4m-1}[2] < v_{4m-1}[2] \qquad \textbf{(8)}$$

Since (v'_{m-2}, v'_{m-1}) is the biggest[1] pair that does not involves v'_{m}, it proves v'_{m} is required to form v_{4m-1}. Besides the biggest pair involving v'_{m} and not v'_{m-1} is (v'_{m-2}, v'_{m}) that is smaller than (v'_{m-2}, v'_{m-1}) since $v'_{m}[2] < v'_{m-1}[2]$. Thus (v'_{m-1}, v'_{m}) is the only pair containing more than 2^{2m-3} ε_2. ■

Theorem 7. *COMPATIBILITY remains NP-complete in the strong sense when the number of symbols is limited to two.*

Proof.

Þ : Assume C and C' are compatible. Putting $2^{2m-3}\varepsilon_2$ at the junction of v'_{m-1} with v'_{m} lets $2^{2(m-1)-4}$ ε_2 available from this pair, and lemma **2** can be reapplied to prove that this rest must be joined to v'_{m-2} to form v_{4m-2} $(0,2^{2m-5})$, etc… This recurrence leads to an order such that symbols ε_1 of each v'_{i} are separated from others by ε_2 sequences, what requires each v_j ($j \leq 3m$) to be assigned to a unique v'_{i}. This assignation induces a 3-partition of the corresponding weighted set X.

Ü : Reciprocally, given a 3-partition of X, a compatibility string can be built like in the previous reduction (section 4.5) ■

5. Conclusions

The NP-completeness of both problems studied in this paper prove the complexity of both multiflow and ordering aspects of the FORMWORK PAIRING PROBLEM (FPP):

- Theorem 3 proves that the one-family case of FPP is NP-hard in the strong sense even with only two days.
- Theorem **6** proves that the pure ordering problem arising when the formwork flow is fixed in a FPP is strongly NP-complete. In other words, once each shuttering is determined to be used on a fixed list of walls, ordering these shutterings on each wall in order to optimize the pairing is non-trivial.

6. References

[B] T. Benoist: *Towards Optimal Formwork Pairing on Construction Sites*. Not published yet.

[GJ75] M.R. Garey and D.S. Johnson: *Complexity results for multiprocessor scheduling under resource constraints*. SIAM J. Comput. 4, 397-411, 1975.

[GJ79] M.R. Garey and D.S. Johnson: *Computers and intractability, a guide to the theory of NP-completeness*. New York: W. H. Freeman, 1979.

[1] In this proof we implicitly name *x[2]+y[2]* the *size* of a pair (x,y).

[K72] R. Karp. *Reducibility among combinatorial problems.* In R.E. Miller and J.W. Thatcher, editors, Complexity of Computer Computations, pages 85--104. Plenum Press, 1972.