Minimizing Places Storage Capacities of a Weighted Event Graph[*]

Olivier Marchetti and Alix Munier-Kordon
Laboratoire d'Informatique de Paris 6 - Pôle ASIM
Université Pierre et Marie Curie
4 place Jussieu, PARIS 75252 Cedex 05, FRANCE
olivier.marchetti@lip6.fr, alix.munier@lip6.fr

10th May 2005

Abstract

The minimization of the places capacities storage of a Weighted Event Graph (in short WEG) is a crucial problem in industrial area such as the design of embedded systems or manufacturing systems. The objective of this paper is to set out an algorithm for solving polynomially this problem. It also computes an initial live marking. Firstly, a lower bound formula for each place capacity storage is expressed. A simple transformation of the initial WEG G into another WEG G_R modelling that the limitation storage of each place of G is equal to these bounds is then detailed. Then, a restriction of the set of feasible initial markings of G_R is made by the analysis of the liveness problem. Finally, we develop a polynomial algorithm which computes an initial live marking of G_R.

Keywords : Petri nets, Liveness, Buffer requirement, Manufacturing.

1 Introduction

Petri nets are a very well suited formalism for the modelling and the analysis of dynamic systems. In this article, our analysis is concentrated on a subclass of Petri nets called Weighted Events Graphs (in short WEG) for which each place p has exactly one input transition t_i and one output transition t_j. Moreover, the number of tokens placed in p after the firing of t_i (*resp.* removed from p for the firing of t_j) is a given integer $w(p)$ (*resp.* $v(p)$). Notices that WEG are conflict free and constitute a very restrictive subclass of Petri nets. However, they may be used for the design and the modelling of manufacturing systems or embedded systems. Transitions represent actions. Tokens correspond to physical products in an assembly line or data exchanged between two processes in embedded systems. In both cases, the capacity storage of any place p, which consists on the maximum number of tokens that p can hold at a time, is proportional to the surface needed to store products or data. In the field of manufacturing systems design, many authors studied the tokens

[*]This research was partialy supported by a partnership CNRS-STMICROELECTRONICS

minimization of an Event Graph under the constraint of the existence of a schedule with a given throughput. In this case, the system is live (*i.e.* every transition may be fired infinitely often) iff there is at least one token in every circuit of the Event Graph. So, the optimization problem is in $\mathcal{NP}$ and several authors develop efficient optimization algorithms to solve it [HP89, Gau90, LPX92] or some interesting variants [GPS02]. On the contrary, the liveness of a WEG is still an open problem despite original attempts to solve it [CWR93]. In a slightly different formalism called Computation Graph [KM66], Karp and Miller has shown that the deadlock decision problem is in $\mathcal{NP}$. From a practical point of view, the liveness problem of a WEG is solved using pseudo-polynomial algorithms [Sau03, TCCS92] which are mainly based upon a transformation of the WEG into an Event Graph introduced in [Mun93, Mun96].

In the field of the design of embedded systems, the high cost of memories implies that the minimization of the capacity storage is crucial. In this area, the Synchronous Data-Flow paradigm (in short SDF) introduced by Lee and Messerschmitt [LM87b, LM87a] is exclusively considered. However, WEG and SDF are equivalent formalisms, since processes of a SDF graph correspond to transitions and arcs to places. In this specific domain, many researches concentrate on the search of periodic schedules with particular features (such as minimal schedule lenght or maximal concurrency schedule) that minimizes the whole amount of memory used [BML99, MBL97, ČP93]. Several authors consider the minimization of places capacities storage of a WEG for a given initial marking. This problem is proved $\mathcal{NP}$-complete even for Event Graphs [Mur96]. Some authors develop heuristics to solve it [Adé97, ALP94]. In [GGD02], the authors formulate by an integer linear program the minimization of the places capacities storage for a periodic schedule of a maximum throughput.

In this paper, we are interested in the minimization of places capacities storage of a WEG in the case that the initial marking is not given. More precisely, let f be an increasing function of the places capacities storage. The question is : is it possible to compute for each place p of a given WEG G, a capacity $M^\star(p)$ and an initial marking $M_0(p)$ such that G is live, the number of tokens of any place p remains anytime bounded by $M^\star(p)$ and $f(M^\star(p_1), \cdots, M^\star(p_m))$ is minimum ? We develop an original (to our best knowledge) polynomial algorithm to solve exactly this problem. Notice that, since the liveness of a WEG is still an open problem, the analysis of the whole problem is subtle. In order to get our result, we consider a sufficient condition of liveness proved in [MMK04].

This paper is structured as follow : Section 2 is devoted to recall the basic concepts of WEG in order to clearly formulate the problem dealt in this paper. In Section 3, we set out some new relevant results about place capacity which form the theoretical basis of our approach. In Section 4, we study the liveness of minimally bounded WEG. Section's 4 results lead us to make some restrictions on the space solution. Then, we devise step by step in Section 5 an algorithm which builds polynomially a live marking. Section 6 displays this algorithm (illustrated by an example) and its generalization. Lastly, we conclude with some perpectives in Section 7.

2 Basic definitions and notations

The aim of this section is to introduce some basic definitions and to express formally the problem tackled in this paper. First, we give the notations and the definition of Weighted Event Graphs. Then, we formulate our problem.

2.1 Basic definitions

Let us consider a Weighted Event Graph $G = (P, T)$ (WEG in short) given by a set of transitions $T = \{t_1, ..., t_n\}$ and a set of places $P = \{p_1, ..., p_m\}$. Every place $p \in P$ is defined between two transitions t_i and t_j and is denoted by $p = (t_i, t_j)$ (see Figure 1 on this page). For any transition $t \in T$, we set :

$$\mathcal{P}^+(t) = \{p = (t, t') \in P, t' \in T\}$$

$$\mathcal{P}^-(t) = \{p = (t', t) \in P, t' \in T\}$$

Arcs (t_i, p) and (p, t_j) are valued by a strictly positive integer denoted respectively by $w(p)$ and $v(p)$ called the marking functions. We also denote by $M_0(p)$ the initial marking of the place p. In order to clearly distinguish the structural aspect from the behavioral aspect, a WEG graph with an initial marking M_0 is called a Marked WEG (MWEG in short) and is denoted by $G = (P, T, M_0)$. Its structure is the WEG $G = (P, T)$.

In this paper, we first focus our attention on the relationships between the structural and behavioural aspects. Then, Section 5 is devoted to the study of strongly connected WEG (*i.e.* for any couple $(t_i, t_j) \in T$, there exists a path in G from t_i to t_j). We will show in Section 6 that strongly connected components of G may be tackled separately.

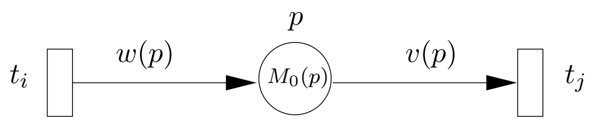

Figure 1: A place $p = (t_i, t_j)$

For any firing of the transition t_i (*resp.* t_j), $w(p)$ (*resp.* $v(p)$) tokens are placed in (*resp.* removed from) the place p. So, if ν_i (*resp.* ν_j) denotes the number of firings of the transition t_i (*resp.* t_j), the number of tokens in the place p is :

$$M(p) = M_0(p) + \nu_i w(p) - \nu_j v(p)$$

For any $\nu \in \mathbb{N}^\star$ and any $t \in T$, $< t, \nu >$ denotes the νth firing of t.

If $w(p) = v(p) = 1$ for any place $p \in P$, then G is an Event Graph. For a sake of simplicity, $gcd(w(p), v(p)) = gcd_p$ denotes the great common divisor of the integers $w(p)$ and $v(p)$ for any place $p \in P$.

2.2 Problem definition

Let $G = (T, P)$ be a WEG. Let $f : \mathbb{N}^m \to \mathbb{N}$ be an increasing function. The problem considered here is to find, for any $p \in P$, a minimum value $M^\star(p) \in \mathbb{N}$ and an initial marking $M_0(p) \leq M^\star(p)$ such that :

- the marked graph $G = (T, P, M_0)$ is live. Moreover, there exists an infinite firings sequence of the transitions such that the number of tokens in any place p remains bounded by $M^\star(p)$;
- $f(M^\star(p_1), \cdots, M^\star(p_m))$ is minimum.

3 Preliminaries

This section reminds and shows some basic useful properties on WEG and MWEG. Firstly, we make a slight restriction on the class of graphs tackled here. After this, we introduce the notion of bounded place and we prove that every bounded place may be replaced by a couple of places. We naturally extend these concepts to define bounded graphs and associated capacity graphs. Then, we recall that any unitary strongly connected graph may be transformed into a normalized graph, which is a subclass of unitary strongly connected graphs such that the marking functions depends on transitions. In the last part, we recall a nice sufficient condition of liveness of normalized graph and we deduce a lower bound of the capacity storage of each bounded place.

3.1 Self-loop places elimination

Definition 3.1. *A place $p = (t_i, t_i)$ is called a self-loop place.*

As we are dealing with WEG, the number of tokens of a self-loop place $p = (t_i, t_i)$ only depends on the number of firings of transition t_i. However, t_i has to be fired infinitely often so that the number of tokens of the whole graph remains bounded on every places. One can see that $w(p) \geq v(p)$ is a necessary condition of liveness for transition t_i and $w(p) \leq v(p)$ is a necessary condition of boundedness of a self-loop place p. So it comes that $w(p) = v(p)$ which induces that the number of tokens on p is always equal to $M_0(p)$. In the same way, $M_0(p) \geq v(p)$ is obviously a necessary condition of liveness of transition t_i. So if all the previous conditions are met then the self-loop place does not prevent at any time t_i to be fired. Moreover, the maximum number of tokens held by a self-loop place p is always equal to its initial marking so we can set $M^\star(p) = M_0(p)$. Since all theses conditions have to be checked for self-loop places, we can limit our study without loss of generality to WEG G free of self-loop places.

3.2 Bounded places and bounded MWEG

Definition 3.2. *A $M^\star$-bounded place is a place $p \in P$ such that the number of tokens of p remains bounded by $M^\star(p) \in \mathbb{N}$. $M^\star(p)$ is called the capacity of place p.*

Let $p = (t_i, t_j) \in P$. There exists a (strict) precedence constraint between $< t_i, \nu_i >$ and $< t_j, \nu_j >$ iff :

Condition 1 $< t_j, \nu_j >$ can be done after $< t_i, \nu_i >$;

Condition 2 $< t_j, \nu_j - 1 >$ can be done before $< t_i, \nu_i >$ but not $< t_j, \nu_j >$.

Lemma 3.1. *A place $p = (t_i, t_j) \in P$ with initial marking $M_0(p)$ models a precedence constraint between the ν_ith firing of t_i and the ν_jth firing of t_j iff :*

$$w(p) > M_0(p) + w(p)\nu_i - v(p)\nu_j \geq \max(w(p) - v(p), 0)$$

Proof. A place $p = (t_i, t_j) \in P$ with initial marking $M_0(p)$ models a precedence constraint between $< t_i, \nu_i >$ and the $< t_j, \nu_j >$ iff Conditions 1 and 2 hold.

1. Condition 1 is equivalent to

$$M_0(p) + w(p)\nu_i - v(p)\nu_j \geq 0$$

2. Condition 2 is equivalent to

$$v(p) > M_0(p) + w(p)(\nu_i - 1) - v(p)(\nu_j - 1) \geq 0$$

Combining these two inequalities, we get the inequality required. □

In the same way, we can derive a condition of precedence between $< t_j, \nu_j >$ and $< t_i, \nu_i >$ due to the storage constraint of place $p = (t_i, t_j)$.

Condition 3 $< t_i, \nu_i >$ can be done after $< t_j, \nu_j >$;

Condition 4 $< t_i, \nu_i - 1 >$ can be done before $< t_j, \nu_j >$ but not $< t_i, \nu_i >$.

Lemma 3.2. *Let $p = (t_i, t_j) \in P$ with initial marking $M_0(p)$. The limitation by an integer $M^\star(p) \geq M_0(p)$ of the capacity storage of place $p = (t_i, t_j) \in P$ induces a precedence constraint between the ν_jth firing of t_j and the ν_ith firing of t_i iff :*

$$v(p) > (M^\star(p) - M_0(p)) + v(p)\nu_j - w(p)\nu_i \geq \max(v(p) - w(p), 0)$$

Proof. A place $p = (t_i, t_j) \in P$ with a capacity bounded by $M^\star(p)$ and $M_0(p)$ initial tokens models a precedence constraint between $< t_j, \nu_j >$ and the $< t_i, \nu_i >$ iff Conditions 3 and 4 hold.

1. Condition 3 is equivalent to

$$M^\star(p) \geq M_0(p) + w(p)\nu_i - v(p)\nu_j$$

We get :

$$(M^\star(p) - M_0(p)) + v(p)\nu_j - w(p)\nu_i \geq 0$$

2. Condition 4 is equivalent to

$$M^\star(p) \geq M_0(p) + w(p)(\nu_i - 1) - v(p)(\nu_j - 1) > M^\star(p) - w(p)$$

So, we get :

$$M^\star(p) + w(p) - v(p) \geq M_0(p) + w(p)\nu_i - v(p)\nu_j > M^\star(p) - v(p)$$

Hence,

$$v(p) > (M^\star(p) - M_0(p)) + v(p)\nu_j - w(p)\nu_i \geq v(p) - w(p)$$

Combining these two inequalities, we get the lemma. □

We deduce the following theorem :

Theorem 3.1. *Any $M^\star$-bounded place p with initial marking $M_0(p) \leq M^\star(p)$ may be replaced by two places $p_1 = (t_i, t_j)$ and $p_2 = (t_j, t_i)$ such that $w(p_1) = v(p_2) = w(p)$, $v(p_1) = w(p_2) = v(p)$, $M_0(p_1) = M_0(p)$ and $M_0(p_2) = M^\star(p) - M_0(p)$ (see Figure 2).*

Proof. By Lemma 3.1 we derive that all precedence constraints between firings of t_i and t_j are modeled by place p_1. By Lemma 3.2 we deduce that all precedence constraints due to the boundedness of p are modeled by place p_2. □

In the following, places p_1 and p_2 are called the associated places of p and we say that p_1 (*resp.* p_2) is the backward place of p_2 (*resp.* p_1).

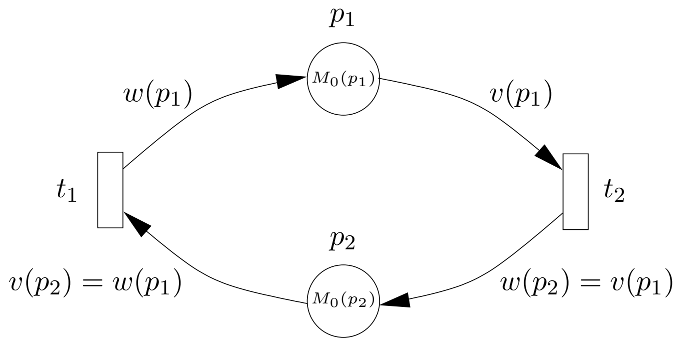

Figure 2: Replacement of an $M^\star$-bounded place $p = (t_1, t_2)$ by $p_1 = (t_1, t_2)$ and $p_2 = (t_2, t_1)$.

In the same way, bounded graphs can be defined as follow :

Definition 3.3. *$G = (T, P, M_0)$ is a bounded graph iff $\forall p \in P$, there exists $M^\star(p) > 0$ such that :*

- *all transitions can be fired infinitely often,*
- *for each place p, the number of tokens remains bounded by $M^\star(p)$.*

3.3 Capacity graphs - Unitary graphs

Firstly, we define the capacity graph obtained by a simple transformation of the WEG places according to Theorem 3.1. Indeed, each $M^\star$-bounded place of graph may be replaced by a couple of places.

Definition 3.4. *Let $G = (T, P, M_0)$ be a MWEG and $\forall p \in P, M^\star(p) \geq M_0(p)$ denotes the capacity bound. The associated capacity graph $G_R = (T, P_R, M_0^\star)$ is the MWEG obtained from G by replacing any place $p \in P$ by two places $(p_1, p_2) \in P_R \times P_R$ following Theorem 3.1.*

We set P_R^1 (resp. *P_R^2) the set of places $p = (t_i, t_j) \in P_R$ such that the corresponding place of P is defined from t_i to t_j* (resp. *t_j to t_i).*

Remark 3.1. *G_R is strongly connected.*

Here, we give a fundamental characterization of capacity graphs which relies on the weight notion of a path. Let us consider a path μ of G defined as a sequence of α places such as $\mu = \{p_1 = (t_1, t_2), p_2 = (t_2, t_3), \ldots, p_\alpha = (t_\alpha, t_{\alpha+1})\}$. The weight $W(\mu)$ of this path is defined as :

$$W(\mu) = \prod_{p \in P \cap \mu} \frac{w(p)}{v(p)}$$

Now, we set out the definition of an important class of WEG called unitary graphs :

Definition 3.5. *A WEG $G = (T, P)$ is a unitary graph iff all its circuits have a unitary weight. Similarly, a MWEG $G = (T, P, M_0)$ is unitary iff its WEG $G = (T, P)$ is unitary.*

Remark 3.2. *One can observes that the replacement of a bounded place p by two places p_1 and p_2 always generates a unitary circuit.*

Assuming that G is a live strongly connected MWEG, it is proved in [Mun93] that the number of tokens remains bounded iff G is a unitary graph. Now, we can give a structural result on bounded graphs :

Theorem 3.2. *Let $G = (T, P, M_0)$ be a MWEG. G is a bounded graph iff there is an associated capacity graph $G_R = (T, P_R, M_0^\star)$ which is unitary and live.*

Proof. We get this theorem by double implications.

• $(\Rightarrow)$: We get this implication by contradiction. Suppose that G is a bounded graph and G_R is not unitary or holds a deadlock whatever $M^\star$.

- First, if G_R is not live whatever $M^\star$ then according to Definition 3.3, it follows that G is not an bounded graph, the contradiction.
- Now, we assume that G is a bounded graph and that G_R is not unitary. In [Mun93], it is stated that a necessary condition of liveness for a MWEG is that all its circuits have a weight greater than one. If G_R is not unitary then there exists a circuit C such that $W(C) > 1$. Then, all its backward places form a circuit C' in G_R. However, we can deduce by Remark 3.2 the following equality :

  $$W(C') = \frac{1}{W(C)}$$

  So it follows that $W(C') < 1$ which implies that G_R is not a live WEG. As this result holds for all $M^\star$, according to Definition 3.3 we deduce that G is not a bounded graph.

• $(\Leftarrow)$: If it exists $M^\star$ such that $G_R = (T, P_R, M_0^\star)$ is live and unitary then according to [Mun93] G_R is a bounded graph so G too.

□

The above theorem defines formally the class of graph with which we are concerned. Without any loss of generality, we assume that G is a bounded graph.

Remark 3.3. *According to Theorem 3.2, the structure of a bounded graph is such that the corresponding capacity graph's structure is unitary. So in the sequel, we focus our attention on graphs whose structure confirms this property.*

Now, we have to consider the liveness problem that arises in our problem's definition.

3.4 Normalization

Normalization of a unitary strongly connected graph G is a transformation of all the marking functions and initial marking values such that the marking functions adjacent to any transition t_i have the same value Z_i. This transformation does not affect the precedence constraints between the firings, so that these two graphs are equivalent. More formally :

Definition 3.6. *A transition t_i is normalized iff there exists $Z_i \in \mathbb{N}^\star$ such that:*

$$\begin{aligned} \forall p \in \mathcal{P}^+(t_i), \quad w(p) &= Z_i \\ \forall p \in \mathcal{P}^-(t_i), \quad v(p) &= Z_i \end{aligned}$$

A WEG G is said normalized iff all its transitions are normalized.

In [MMK04], it is stated that any strongly connected unitary MWEG can be polynomially transformed into an equivalent normalized MWEG by modifying marking functions and initial markings.

Consequence 3.1. *As we are concerned with bounded graph and according to Remark 3.1 and Theorem 3.2, capacity graphs which are tackled here can be normalized. It follows that bounded graphs can be normalized too.*

In the rest of the paper, we assume that all graphs which are dealt here are normalized.

3.5 Lower bound on the capacity storage of places

We present here a lower bound on the capacity storage of bounded places of G. Firstly, we remind some results proved in [MMK04]. Then we prove these lower bounds formally.

The next lemma allows us to limit the values of initial markings of a MWEG :

Lemma 3.3 ([MMK04]). *The initial marking $M_0(p)$ of any place $p = (t_i, t_j)$ may be replaced by $M_0'(p) = \left\lfloor \frac{M_0(p)}{gcd_p} \right\rfloor .gcd_p$ without any influence on the precedence constraints induced by p.*

In the rest of the paper, we assume that the initial marking of any place p is a multiple of gcd_p.

Theorem 3.3 ([MMK04]). *Let G be a normalized MWEG. G is live if for all circuit C of G :*

$$\sum_{p \in C \cap P} M_0(p) > \sum_{p \in C \cap P} (v(p) - gcd_p)$$

It is shown in [MMK04] that this condition is also necessary for any circuit with two transitions :

Theorem 3.4 ([MMK04]). *Let C be a normalized circuit composed by two places p_1 and p_2 and two different transitions. C is live iff*

$$M_0(p_1) + M_0(p_2) > v(p_1) + v(p_2) - 2.gcd_{p_1}$$

It is now possible to determine a lower bound $M^\star_{min}(p)$ of the capacity storage $M^\star(p)$ of a place p as follows :

Theorem 3.5. *Let p be a $M^\star$-bounded place. Setting $M^\star_{min}(p) = w(p) + v(p) - gcd_p$, we get $M^\star(p) \geq M^\star_{min}(p)$.*

Proof. By Theorem 3.1, p may be replaced by a circuit with two places p_1 and p_2. By Lemma 3.3, inequality of Theorem 3.4 can be refined such that :

$$M_0(p_1) + M_0(p_2) \geq v(p_1) + v(p_2) - gcd_{p_1}$$

By definition of p_1 and p_2, $M^\star(p) = M_0(p_1) + M_0(p_2)$ and $v(p_1) = v(p)$, $v(p_2) = w(p)$ and $gcd_{p_1} = gcd_{p_2} = gcd_p$. So it comes that :

$$M^\star(p) \geq w(p) + v(p) - gcd_p$$

Setting $M^\star_{min}(p) = w(p) + v(p) - gcd_p$, we get the theorem. □

All these results allow us to strengthen Theorem 3.2 :

Theorem 3.6. *Let $G = (T, P, M_0)$ be a live MWEG. G is a bounded graph iff the associated capacity graph's structure $G_R = (T, P_R)$ is unitary.*

Proof. As the first implication of the proof of Theorem 3.2 still holds, we just need to show the converse proposition. More precisely, we have to show that if $G = (T, P, M_0)$ is a live MWEG such that its capacity graph's structure is unitary then there exists an infinite sequence of firing such that for each place p the number of tokens remains bounded by a value $M^\star(p)$. Firstly, according to Consequence 3.1, as $G_R = (T, P_R)$ is unitary, it can be normalized. We consider that $G_R = (T, P_R)$ is normalized. In order to show that G is a bounded graph, we have to show that there exists an initial marking for each place of P^2_R such that G_R is live. By putting $B^\star(p) = \left\lceil \frac{\sum_{p \in P} M^\star_{min}(p)}{gcp_p} \right\rceil .gcd_p$ tokens on each place p of P^2_R, we ascertain that G_R is live (notice that since G_R is normalized and each place $p_2 \in P^2_R$ partially models a place $p \in P$, we have $gcd_{p_2} = gcd_p$ and therefore the value $B^\star(p)$ only depends on the place $p \in P$). Indeed, according to Theorem 3.5, it can be claim that all circuit of size 2 of G_R is live. In the same way, the sufficient condition of liveness of Theorem 3.3 is checked for all the circuit except perhaps those which are solely made of place of P^1_R. However, we have assumed that $G = (T, P, M_0)$ is a live MWEG, so all circuits of G are live. So, we have found an initial marking such that G_R is live. It follows that G is a bounded graph with capacities $M^\star(p) = M_0(p) + B^\star(p)$, $\forall p \in P$. □

4 Restriction and simplification of the problem

In this Section, we define the minimum capacity graph, which is obtained by transforming every bounded place of G into a couple of places with the sum of initial tokens equal to the lower bound shown previously. Then, we express a sufficient condition of liveness for this type of MWEG. Lastly, we restrict the problem to the determination of a live marking of the minimum capacity graph.

4.1 Definition

Definition 4.1. *The minimum capacity graph denoted by $G_R^{min} = (T, P_R, M_0^\star)$ of a MWEG $G = (P, T, M_0)$ with $\forall p \in P,\ M_0(p) \le M_{min}^\star(p)$ is the capacity graph which is built in accordance with building rules of Theorem 3.1 with $M^\star = M_{min}^\star$.*

According to Consequence 3.1, the minimum capacity graph is a normalized MWEG so the liveness property defined below still works.

Clearly, graph G_R^{min} is live iff there exists an infinite sequences of firings of G such that, for any place $p \in P$, the instantaneous marking of p remains bounded by $M_{min}^\star(p)$. In the following, we study the liveness of G_R^{min}.

4.2 A sufficient condition of liveness

In order to establish the sufficient condition of liveness for any minimum capacity graph, we first provide a necessary condition of liveness which allow us to restrict the set of MWEG dealt subsequently such as we have done it for the self-loop places case.

Definition 4.2. *Let $G = (P, T)$ be a WEG. Two places $p = (t_i, t_j)$ and $q = (t_i, t_j)$ are said parallel. A generic WEG is a WEG without parallel places. So by merely deleting parallel places of WEG, we can build its generic WEG.*

Lemma 4.1. *Let $G = (P, T, M_0)$ be a MWEG with $\forall p \in P,\ M_0(p) \le M_{min}^\star(p)$. If the minimum capacity graph $G_R^{min} = (T, P_R, M_0^\star)$ is live then for any couple (p, q) of parallel places we have $M_0(p) = M_0(q)$.*

Proof. This proof is get by contradiction. Assume that there exists two parallel places $p = (t_i, t_j)$ and $q = (t_i, t_j)$ in G such as $M_0(p) \le M_{min}^\star(p)$, $M_0(q) \le M_{min}^\star(q)$, $M_0(p) > M_0(q)$ and $G_R^{min} = (T, P_R, M_0^\star)$ is live. It comes that $G_R^{min} = (T, P_R, M_0^\star)$ holds a backward place p' (*resp.* q') associated to p (*resp.* q) with $M_0(p') = M_{min}^\star(p) - M_0(p)$ (*resp.* $M_0(q') = M_{min}^\star(q) - M_0(q)$). Since G is normalized then $w(p) = w(q) = v(p') = v(q')$, $v(p) = v(q) = w(p') = w(q')$ and $gcd_p = gcd_q = gcd_{q'} = gcd_{p'}$. We deduce that $M_{min}^\star(p) = M_{min}^\star(q)$. Let us consider the initial marking of circuit $C = \{q = (t_i, t_j), p' = (t_j, t_i)\}$:

$$\begin{array}{rcl} M_0(q) + M_0(p') & = & M_0(q) + M_{min}^\star(p) - M_0(p) \\ & < & M_0(q) + M_{min}^\star(p) - M_0(q) \\ & < & M_{min}^\star(p) \\ & < & v(q) + v(p') - gcd_q \end{array}$$

So according to theorem 3.4, this circuit is not live. We get a contradiction. □

Remark 4.1. *According to lemma 4.1 and to lemma 3.1, since parallel places with the same number of initial tokens model exactly the same constraints, it follows that given a WEG which holds some parallel places there is an initial live marking for its minimum capacity graph if and only if there is an initial live marking for the minimum capacity graph of its generic WEG. So without any loss of generality, we focus our study on generic WEG in the rest of the paper.*

Theorem 4.1. *Let us suppose that, for any circuit C of G_R^{min},*

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0(p) > \sum_{p \in C \cap P_R} (v(p) - gcd_p)$$

then, G_R^{min} is live.

Proof. Let us consider a circuit C of G_R^{min}. We have to consider two cases :

- If the number of places in C is equal to 2 then C involves two transitions denoted here by t_i and t_j. Assume moreover that there is only one place p between them in G. It follows that, by contruction of G_R^{min} and by theorem 3.4, C is live.

  Now, since $\sum_{p \in C \cap P_R} M_0(p) = w(p) + v(p) - gcd_p$

  we deduce $\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0(p) > \sum_{p \in C \cap P_R} (v(p) - gcd_p)$

- Otherwise, in accordance with G_R^{min}'s building rules, there exists a circuit C' of G_R^{min} such that, to any place $p = (t_i, t_j)$ of C corresponds a backward place $p' = (t_j, t_i)$ in C'. Since G_R^{min} is normalized then for any couple of places $(p, q) \in P_R \times P_R$ which both share the same transitions, we have $M^\star_{min}(p) = M^\star_{min}(q)$. So the number of tokens in places from $C \cup C'$ verifies :

$$\begin{aligned} N_{C \cup C'} &= \sum_{p \in C \cap P_R} M^\star_{min}(p) \\ &= \sum_{p \in C \cap P_R} (w(p) + v(p) - gcd_p) \\ &= \sum_{p \in C \cap P_R} M_0(p) + \sum_{p \in C' \cap P_R} M_0(p) \end{aligned}$$

  As G_R^{min} is normalized, by Theorem 3.3, C is live if

$$\sum_{p \in C \cap P_R} M_0(p) > \sum_{p \in C \cap P_R} (v(p) - gcd_p)$$

  In the same way, C' is live if

$$\sum_{p \in C' \cap P_R} M_0(p) > \sum_{p \in C' \cap P_R} (v(p) - gcd_p)$$

  Since $\sum_{p \in C' \cap P_R} (v(p) - gcd_p) = \sum_{p \in C \cap P_R} (w(p) - gcd_p)$

and $\sum_{p \in C' \cap P_R} M_0(p) = \sum_{p \in C \cap P_R} (w(p) + v(p) - gcd_p) - \sum_{p \in C \cap P_R} M_0(p)$, the condition becomes :

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0(p)$$

□

4.3 Problem restriction

In next Section, we consider the more restrictive version of our prime problem (*see.* sub-section 2.2 on page 3) for which $M^\star(p) = M^\star_{min}(p)$ for each place p of G. In the sequel, when we talk about capacity graph, as opposed to Definition 4.1 and as our objective is to build a marking, we only refer to its structure which is common to all capacity graphs and we omit the marking's definition.

We develop a polynomial algorithm which computes a live initial marking of a capacity graph such that for each couple of place $p = (t_i, t_j)$ and $p' = (t_j, t_i)$ we get $M_0(p)+M_0(p') = M^\star_{min}(p)$. As we are seeking a marking such that the graph is live and minimally bounded, initially $G_R^{min} = (T, P_R)$ just denotes the structure of the capacity graph. For any place p of G_R^{min}, the two possible initial markings considered by this algorithm are $v(p)$ or $v(p)-gcd_p$.

More formally, the algorithm is a constructive proof of the following theorem :

Theorem 4.2. *Let $G_R^{min} = (T, P_R)$ be the capacity graph's structure of a unitary WEG $G = (T, P)$. An initial live marking of G_R^{min} can be built in polynomial time such that, for any place $p \in P_R$, $M_0(p) \in \{v(p), v(p) - gcd_p\}$.*

5 Towards a polynomial algorithm for the construction of a live marking under restriction of minimum capacity storage

In this Section, we only consider the problem of building for strongly connected bounded graph an initial marking which is live and minimally bounded. The general case will be treated in the forthcoming section.

Theorem 5.1. *Let $G_R^{min} = (T, P_R)$ be the capacity graph's structure of a strongly connected unitary graph $G = (T, P)$. An initial live marking of G_R^{min} can be built in polynomial time such that, for any place $p \in P_R$, $M_0(p) \in \{v(p), v(p) - gcd_p\}$.*

Firstly, we transform G_R^{min} into an associated graph $\mathcal{G}$ in order to simplify the presentation of the algorithm. We also translate the sufficient condition of liveness of G_R^{min} into a property of the labeling of arcs of $\mathcal{G}$. We deduce from this last property the termination of the algorithm. Then, we describe the initialization and the iterative step.

5.1 Definition of the associated graph $\mathcal{G}$

Let G be a strongly connected normalized unitary WEG and G_R^{min} its associated capacity graph's structure. In order to simplify the proof's writing, we define the associated oriented graph $\mathcal{G} = (T, U)$ as follows :

1. vertices of $\mathcal{G}$ are transitions of G;
2. any place $p = (t_i, t_j)$ of P_R is associated with an arc u_p from t_i to t_j. Moreover, in order to differentiate places of P_R^1 from those of P_R^2, we use on figures a solid style (*resp.* dotted style) to depict arcs of U associated with P_R^1 (*resp.* P_R^2).

By hypothesis, the initial marking of any place $p \in P_R$ verifies $M_0(p) \in \{v(p), v(p) - gcd_p\}$. So, to any initial marking M_0 of G_R^{min}, we associate a label l of the arcs of $\mathcal{G}$ defined as :

$$\begin{cases} l(u_p) = 0 & \text{if } M_0(p) = v(p) - gcd_p \\ l(u_p) = 1 & \text{if } M_0(p) = v(p) \end{cases}$$

Conversely, to any label of $\mathcal{G}$, an initial marking of G_R^{min} can be associated.

Property 5.1. *If M_0 is a live marking of G_R^{min}, then for any couple of places $(p_1, p_2) \in P_R^1 \times P_R^2$ associated to a same place p of G, $l(u_{p_1}) + l(u_{p_2}) = 1$*

Proof. By definition of G_R^{min}, $M_0(p_1) + M_0(p_2) = M_{min}^\star(p) = w(p) + v(p) - gcd_p$. Since $M_0(p_1) \in \{v(p), v(p) - gcd_p\}$ and $M_0(p_2) \in \{w(p), w(p) - gcd_p\}$, we deduce the property. □

Lemma 5.1. *Let l be a label of $\mathcal{G}$ which verifies Property 5.1. If every circuit $\mathcal{C}$ of $\mathcal{G}$ has at least one arc x and one arc y such that $l(x) = 0$ and $l(y) = 1$, then the associated marking M_0 of G_R^{min} is live.*

Proof. Let us consider a circuit $\mathcal{C}$ of $\mathcal{G}$ and let C be the corresponding circuit of G_R^{min}. If l verifies the condition of the lemma, then

$$|\mathcal{C}| > \sum_{p \in C \cap P_R} l(u_p) > 0$$

where $|\mathcal{C}|$ is the number of arcs in $\mathcal{C}$. We get :

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0(p) > \sum_{p \in C \cap P_R} (v(p) - gcd_p)$$

By Theorem 3.3, we deduce that the associated marking M_0 of G_R^{min} is live. □

One can notice that this problem seems to be closely related to the well-know $\mathcal{NP}$-Complete problem FEEDBACK ARC SET [Kar72].

5.2 Termination of the algorithm

Let U_0 be the set of null labeled arcs of $\mathcal{G}$ and $\mathcal{G}_0 = (T, U_0)$. Likewise, let U_0^Δ be the subset of dotted arcs from U_0 and $\mathcal{G}_0^\Delta = (T, U_0^\Delta)$.

Theorem 5.2. *If $\mathcal{G}_0$ is a Directed Acyclic Graph (DAG in short) then the corresponding marking M_0 of G_R^{min} is live.*

Proof. Let us suppose that G_R^{min} is not live for a marking M_0. Since l verifies Property 5.1, by Lemma 5.1, there exists a circuit $\mathcal{C}$ of $\mathcal{G}$ such that $l(x) = 0$ for any arc x of $\mathcal{C}$. $\mathcal{C}$ is included in $\mathcal{G}_0$, which is not a DAG. □

5.3 Initialization of $\mathcal{G}_0$

Let us consider a spanning out-tree $\mathcal{T}$ of G (*i.e.* $\mathcal{T}$ is a partial subgraph of G which is an out-tree and connecting every vertex of G). $\mathcal{T}$ exists since G is strongly connected. We build an initial label of $\mathcal{G}$ as follows :

- If u_p is a solid arc, then $l(u_p) = 1$ iff the corresponding place $p \in P$ of G is in $\mathcal{T}$.
- If u_p is a dotted arc, then $l(u_p) = 1$ iff the backwards solid arc $u_{p'}$ has its label $l(u_{p'}) = 0$.

Property 5.2. *$\mathcal{G}_0$ does not hold any dotted circuit.*

Proof. Assume that $\mathcal{G}_0$ holds a dotted circuit $\mathcal{C}$. According to the building's rules of the labels, Property 5.1 holds. So, the backwards circuit $\mathcal{C}'$ of $\mathcal{C}$ is composed by solid arcs and $\mathcal{C}'$ is included in $\mathcal{T}$, which is impossible. □

Now, since $\mathcal{G}_0$ does not hold dotted circuits :

Consequence 5.1. *Initially, all circuits of $\mathcal{G}_0$ are either solely made of solid arcs or by solid and dotted arcs.*

5.4 Breaking circuits of $\mathcal{G}_0$

The idea of the algorithm is to compute a label of $\mathcal{G}$ such that $\mathcal{G}_0$ has no circuit. By Theorem 5.2, the corresponding initial marking of G_R^{min} is live.

At this step of the algorithm, one can imagine that $\mathcal{G}_0$ is (still) not a DAG, so we have to devise an algorithm which breaks all circuits of $\mathcal{G}_0$.

Theorem 5.3. *If $\mathcal{G}_0$ is free of dotted circuits and contains a circuit $\mathcal{C}$, then there is at least one solid arc of $\mathcal{C}$ which can be labeled by 1 without creating dotted circuits in $\mathcal{G}_0$.*

Proof. We prove it by contradiction. Suppose that $\mathcal{G}_0$ is free of dotted circuits. Let $\mathcal{C}$ be a circuit such that no solid arc of $\mathcal{C}$ may be labeled by 1 without creating a dotted circuit by appending to $\mathcal{G}_0$ the corresponding dotted backward arc. Notice that removing a solid arc from $\mathcal{G}_0$ cannot create a solid circuit in $\mathcal{G}_0$. In this case, it follows that there were already a dotted circuit in $\mathcal{G}_0$ (*see.* Figure 3 on the following page), the contradiction. □

The algorithm will remove arcs from $\mathcal{G}_0$ following Theorem 5.3. So, the number of solid arcs of $\mathcal{G}_0$ is strictly decreasing at each step of the algorithm. The algorithm stops when $\mathcal{G}_0$ has no more circuits.

6 Algorithms and an example

In this section, we sum up the algorithm for the strongly connected case and we express its complexity. Then, we will see a basic example. Finally, we extend these results to all bounded graphs.

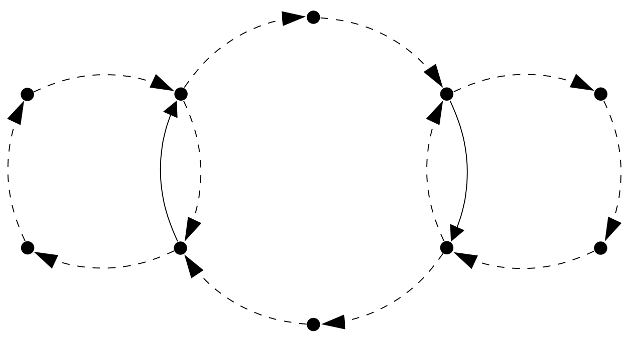

Figure 3: At the center of the figure, we have the considered circuit $\mathcal{C}$. If, it is not possible to change any labels of solid arcs without creating some new dotted circuits, then there were previously in $\mathcal{G}_0$ a dotted circuit (cf. the dotted outside circuit).

6.1 Algorithm for a strongly connected graph

Firstly, one can observes that the algorithm computes for each place p an initial marking $M_0(p) \in \{v(p)-gcd_p, v(p)\}$. As the normalization operation is an increasing linear function of initial place's attributes (*i.e.* $w(p)$,$v(p)$, $M_0(p)$ and as $gcd(k.w(p), k.v(p)) = k.gcd_p$), the normalization step is not necessary.

1. Build an out-tree $\mathcal{T}$ of the initial graph G using DIJKSTRA's algorithm. For any arc of $\mathcal{T}$, label the corresponding solid arc of $\mathcal{G}$ by 1. For any arc of $G - \mathcal{T}$, label the corresponding solid arc of $\mathcal{G}$ by 0. Dotted arcs of $\mathcal{G}$ are labeled following Property 5.1.
2. Extract from $\mathcal{G}$ the subgraph $\mathcal{G}_0$.
3. Check with the DEPTH-FIRST SEARCH algorithm that $\mathcal{G}_0$ is a DAG :
   (a) If $\mathcal{G}_0$ is a DAG then the associated initial marking of G_R^{min} is live (*see.* Theorem 5.2). Go to STEP 4.
   (b) If $\mathcal{G}_0$ holds a circuit $\mathcal{C}$ then let S be the set of solid arcs from $\mathcal{C}$.
      i. Select an arc $u_s \in S$ and denote by u_d its backwards dotted arc.
      ii. If the dotted subgraph $\mathcal{G}_0^{\Delta'} = (T, U_0^{\Delta} \cup \{u_d\})$ is a DAG, then u_s may be labeled by 1 : set $U_0 = U_0 \cup \{u_d\} - \{u_s\}$ and go back to STEP 3.
      iii. Otherwise, set $S = S - \{u_s\}$ and go to STEP 3(b)i.
4. Build an initial marking of G_R^{min} as follow :
   for each arc u_p of $\mathcal{G}$, set $M_0(p) = v(p) - gcd_p$ if $l(u_p) = 0$ else set $M_0(p) = v(p)$.

Lemma 6.1. *The complexity of this algorithm is in $\mathcal{O}(m^2)$.*

Proof. The complexity of DIJKSTRA and DEPTH-FIRST SEARCH algorithms are in $\mathcal{O}(m)$. At each step of the algorithm, a solid arc is selected and we have to check that if this arc can be safely replaced by its backward dotted arc by using the DEPTH-FIRST SEARCH algorithm. As the number of iterations of STEP 3 is in $\mathcal{O}(m)$ then it follows that the whole complexity of the algorithm is in $\mathcal{O}(m^2)$. □

6.2 A basic example

Now, we apply the algorithm on a small example. The initial WEG is depicted by Figure 4 a) on the current page. Figure 4 b) shows the capacity graph's structure.

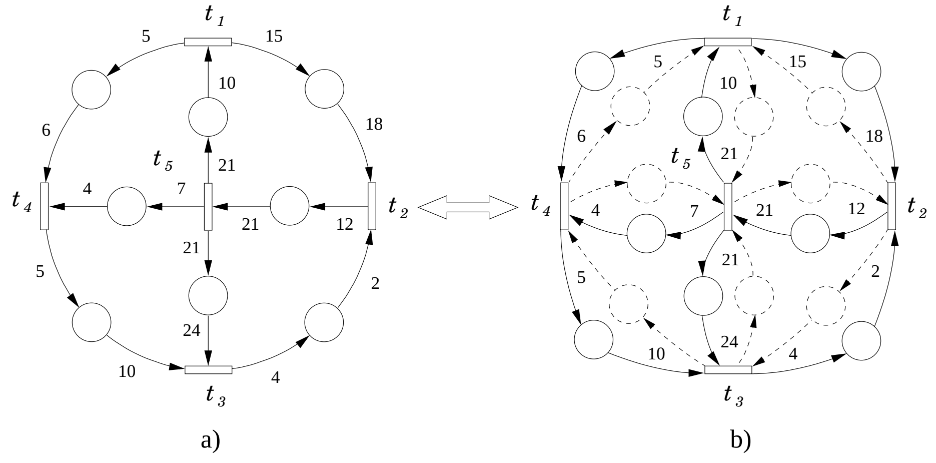

Figure 4: Let us consider the WEG depicted above on the left. On the right hand side, the capacity graph is presented. Now, we are looking for a live marking on this capacity graph.

According to STEP 1, the algorithm builds a spanning out-tree $\mathcal{T}$ from vertex t_5 for example. The arcs of this spanning out-tree $\mathcal{T}$ are labeled by 1. The resulting graph is shown by Figure 5 c) on this page. Then, the labels for the associated graph $\mathcal{G}$ (as shown by Figure 5 d)) are computed as defined in Section 5.

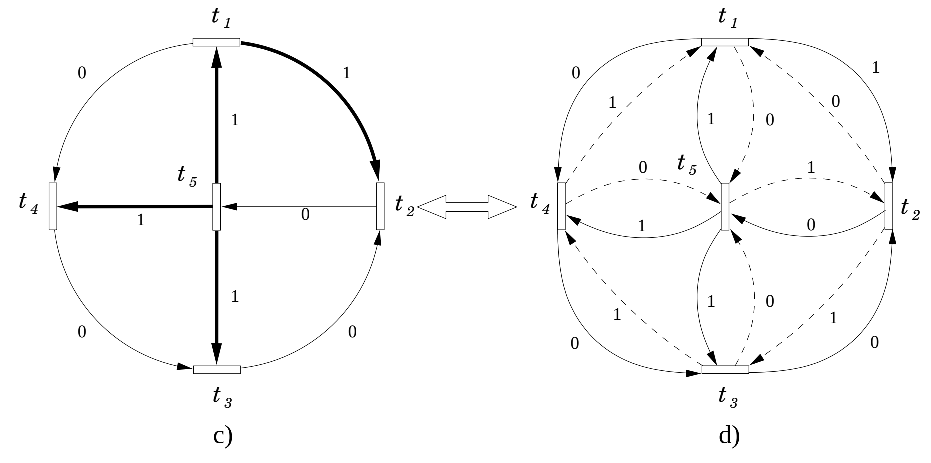

Figure 5: The initialization step is made on Figure c). Figure d) depicts the associated graph with this initial label.

The subgraph $\mathcal{G}_0$ extracted from $\mathcal{G}$ is represented by Figure 6 e) on the next page. Then at STEP 3, the algorithm checks that $\mathcal{G}_0$ is a DAG. Here, the algorithm finds a circuit which is depicted with thick arcs and we set out it on Figure 6 e). There is several candidate arcs in order to break this circuit. So we choose to break the circuit by "reversing" the arc (t_1, t_4) as shown on the right hand side. At the next iteration, the algorithm attests that the modified graph $\mathcal{G}_0$ is a DAG (Figure 6 f) on this page). So we achieve our goal and the algorithm stops.

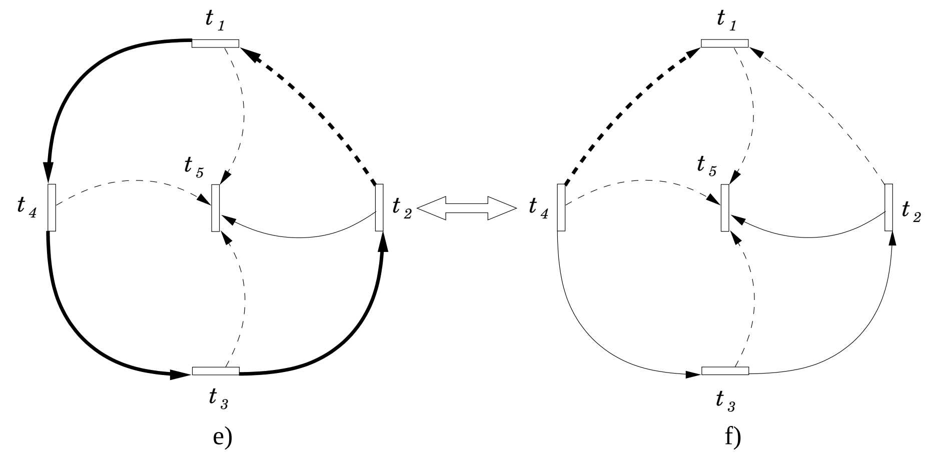

Figure 6: On the right hand side, a circuit of $\mathcal{G}_0$ is displayed. Figure f) shows the resulting graph at the end of the first iteration of STEP 3.

The whole label for this graph can be easily deduced according to Property 5.1. It is given on the left hand side of Figure 7 on the current page. Lastly, according to the definition of the associated graph, we can computes a live initial marking for our WEG which is minimally bounded for each place.

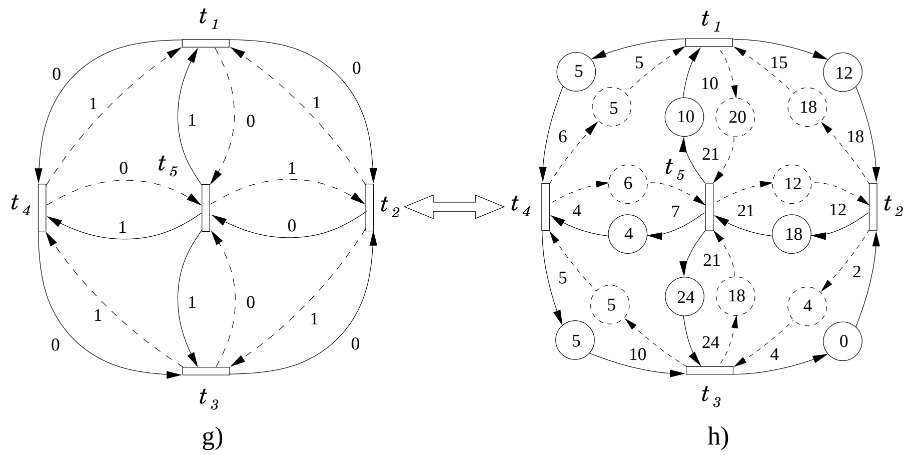

Figure 7: On Figure g), the whole label is shown for the associated graph $\mathcal{G}$. Then, we have depicted the initial marking for the G_R^{min}. This initial marking is live.

6.3 Generalization

We extend the first algorithm in order to treat all bounded graphs. In the sequel, if G contains k strongly connected components then they will be denoted by $C_1, \dots, C_k$. For this purpose, we need to introduce a new definition :

Definition 6.1. *An independent place $p = (t_x, t_y)$ is a place such that :*

$$(t_x, t_y) \in C_i \times C_j \ , (i, j) \in \{1, \dots, k\}^2, i \neq j$$

The set of independent places of G is denoted by $P_I \subseteq P$. In the same way, $P_I^a \subseteq P_R$ denotes the set of places of P_R associated with an independant place.

The following algorithm is the generalization of the first one :

1. Compute all strongly connected components of graph G using the Depth-First Search's algorithm. Determine the set P_I of independant places of G.
2. Compute using the previous algorithm a marking for each strongly connected component of G.
3. For each place $p \in P_I^a$, if $p \in P_R^1$ set $M_0(p) = v(p)$ else $M_0(p) = v(p) - gcd_p$.

Theorem 6.1. *This algorithm computes a live initial marking for G_R^{min}.*

Proof. This proof is get by contradiction. Assume that G holds k strongly connected components and all places are marked according to the algorithm computation. If this initial marking is not live then, by Theorem 4.1, there is in the capacity graph G_R^{min} a circuit C where each place p is marked by $v(p) - gcd_p$ tokens. We consider two cases :

- First case : If C does not involve any associated place of any component C_i then the circuit is solely made of places of P_I^a. As G does not hold a circuit which is solely made of places from P_I then we deduce that C holds at least 1 place of P_R^1 and 1 place of P_R^2. However according to the algorithm all places $p \in P_R^1 \cap P_I^a$ are initially marked with $v(p)$ tokens. So, we get a contradiction.
- Second case : If C contains some associated places from at least one strongly connected component then according to Theorem 5.1 we can deduce that C holds at least two places of P_I^a. We have now two sub-cases :
  - All places of $C \cap P_I^a$ are in P_R^2. If all those places belong to P_R^2 then the backward circuit C' defines a new strongly connected component in G, a contradiction.
  - So, there is at least one place of $C \cap P_I^a$ which is in P_R^1. Then this place was not marked according to our algorithm.

□

Theorem 6.2. *The complexity of the whole approach is in $\mathcal{O}(n.m^2)$*

Proof. The number of computations of the global algorithm mainly depends on the use of the DEPTH-FIRST SEARCH's algorithm and on the algorithm for the strongly component case. The first and the third steps of the algorithm are both in $\mathcal{O}(m)$. As $|T| = n$, the number of strongly component of a graph G is in $\mathcal{O}(n)$. However, according to Lemma 6.1, the number of computations of the strongly component case is in $\mathcal{O}(m^2)$. It follows that the number of computations performed at the second step is in $\mathcal{O}(n.m^2)$. So, the theorem holds. □

7 Conclusion

We have been concerned in this paper with a major problem of industrial area. This article has proposed a new approach for solving this problem by introducing some new constraints. These restrictions facilitate the design of a polynomial algorithm that computes an initial marking for WEG which is live and minimally bounded. To our knowledge, all computational solutions that have been devised until now were heuristics or pseudo-polynomial algorithms. So our polynomial algorithm may be used for treating large size instances and computing optimal results.

We plan to study in future-work the relationship between the throughput of a MWEG and the capacity constraints over places.

References

[Adé97] M. Adé, *Data memory minimization for synchronous data flow graphs emulated on dsp-fpga targets*, Ph.D. thesis, Université Catholique de Louvain, 1997.

[ALP94] M. Adé, R. Lauwereins, and J. A. Peperstraete, *Buffer memory requirements in dsp applications*, IEEE 5th International Workshop on Rapid System Prototyping (Grenoble, France), 1994, pp. 108–123.

[BML99] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, *Synthesis of embedded software from synchronous dataflow specifications*, Journal of VLSI Signal Processing (1999), no. 21, 151–166.

[ČP93] M. Čubrić and P. Panangaden, *Minimal memory schedules for dataflow networks.*, CONCUR '93, 4th International Conference on Concurrency Theory, Lecture Notes in Computer Sciences, vol. 715, 1993, pp. 368–383.

[CWR93] P. Chrzastowski-Wachtel and M. Raczunas, *Liveness of weighted circuits and the diophantine problem of frobenius.*, FCT, 1993, pp. 171–180.

[Gau90] S. Gaubert, *An algebraic method for optimizing resources in timed event graphs*, 9th conference on Analysis and Optimization of Systems, Lecture Notes in Computer Sciences, vol. 144, 1990, pp. 957–966.

[GGD02] R. Govindarajan, G. Gao, and P. Desai, *Minimizing memory requirements in rate-optimal schedule in regular dataflow networks*, Journal of VLSI Signal Processing **31** (2002), no. 3.

[GPS02] A. Giua, A. Piccaluga, and C. Seatzu, *Firing rate optimization of cyclic timed event graphs.*, Automatica **38** (2002), no. 1, 91–103.

[HP89] H. Hillion and J-M. Proth, *Performance evaluation of a job-shop system using timed event graph*, IEEE transactions on automatic control **34** (1989), no. 1, 3–9.

[Kar72] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, 1972, pp. 85–103.

[KM66] R. M. Karp and R. E. Miller, *Properties of a model for parallel computations: Determinacy, termination, queueing*, SIAM **14** (1966), no. 63, 1390 – 1411.

[LM87a] E. A. Lee and D. G. Messerschmitt, *Static scheduling of synchronous data flow programs for digital signal processing*, IEEE Transaction on Computers **C-36** (1987), no. 1, 24–35.

[LM87b] ———, *Synchronous data flow*, IEEE Proceedings of the IEEE **75** (1987), no. 9.

[LPX92] S. Laftit, J-M. Proth, and X. Xie, *Optimization of invariant criteria for event graphs*, IEEE Transactions on Automatic Control **37** (1992), no. 5, 547–555.

[MBL97] P. K. Murthy, S. S. Bhattacharyya, and E. A. Lee, *Joint minimization of code and data for synchonous dataflow programs*, Journal of Formal Methods in System Design **11** (1997), no. 1.

[MMK04] O. Marchetti and A. Munier-Kordon, *A sufficient condition for the liveness of weighted event graphs*, Research report, LIP6, Laboratoire d'Informatique de Paris 6, 2004, available at ftp://asim.lip6.fr/pub/reports/2004/rp.lip6.2004.marchetti.pdf.

[Mun93] A. Munier, *Régime asymptotique optimal d'un graphe d'événements temporisé : application à un problème d'assemblage*, RAIRO (1993).

[Mun96] A. Munier, *The basic cyclic scheduling problem with linear precedence constraints.*, Discrete Applied Mathematics **64** (1996), no. 3, 219–238.

[Mur96] P. M. Murthy, *Scheduling techniques for synchronous and multidimensional synchronous dataflow.*, Ph.D. thesis, University of California at Berkeley, 1996.

[Sau03] N. Sauer, *Marking optimization of weighted marked graphs*, Discrete Event Dynamic Systems **13** (2003), no. 3, 245–262.

[TCCS92] E. Teruel, P. Chrzastowski, J. M. Colom, and M. Silva, *On weighted t-systems*, Application and Theory of Petri Nets 1992 (K. Jensen, ed.), vol. 616, Springer, 1992, pp. 348–367.