Automazione I

16 Gennaio 2015

Esercizio 1

Si consideri un sistema di automazione industriale in cui, a livello di coordinamento, è necessario portare a termine i seguenti task periodici:

1. ogni 12 t.u., una lamiera viene trasferita in zona di lavorazione impiegando 2 t.u.
2. ogni 3 t.u., una lamiera viene forata impiegando 1 t.u.
3. ogni 5 t.u., una lamiera viene sagomata impiegando 2 t.u.

Si ipotizzi che tutti i task siano indipendenti l'uno dall'altro e debbano essere serviti in maniera hard real time. Ai task suddetti si aggiunge un task aperiodico di controllo di qualità della lavorazione. L'activation time della prima occorrenza di tale task aperiodico è fissato all'istante $a_4(1) = 16$ t.u., con un computation time pari a $C_4(1) = 3$ t.u., una deadline relativa pari a $d_4(1) = 15$ t.u., e deve essere servito in maniera soft real time.

Si chiede di rispondere ai seguenti punti.

- Verificare se sussiste la condizione necessaria per l'ammissibilità del problema.
- Verificare se sussiste almeno una condizione sufficiente che garantisca la schedulabilità dei task hard real time tramite RMPO.
- Mostrare il risultato dello scheduling usando RMPO per i task periodici e un servizio in background di tipo FIFO per il task soft real time.
- Relativamente al problema di scheduling dei soli task periodici, verificare se il processore risulta essere **completamente utilizzato** rispetto all'algoritmo di scheduling RMPO.
- Determinare se il task soft real time viene eseguito entro la deadline. In caso negativo, verificare, a parità di computation time (3 t.u.) e di deadline relativa (15 t.u.), se esiste un istante di attivazione $a_4(1)$ tale che il task soft real venga eseguito entro la deadline.

Esercizio 2

In una cella di produzione, il controllore C_r di un robot richiede informazioni al controllore C_{mu} di una macchina utensile sullo stato del processo in lavorazione. Lo scambio di messaggi avviene su due canali separati mono-direzionali e dedicati. Ogni volta che C_r è pronto ad inviare una richiesta, la invia sul canale 1 e si mette in attesa di risposta da parte di C_{mu}. Dopo averla ricevuta, C_r torna nello stato di pronto ad inviare. Quando il controllore C_{mu} è pronto a ricevere la richiesta, la riceve e la elabora. Al termine, invia la risposta a C_r sul canale 2 e ritorna nello stato di pronto a ricevere. All'inizio, C_r e C_{mu} sono rispettivamente nello stato di pronto a inviare e pronto a ricevere.

Modellizzare questo protocollo di comunicazione mediante una rete di Petri, eventualmente temporizzata, in modo che il funzionamento si possa ripetere all'infinito.

Esercizio 3

Per la rete di Petri in Fig. 1 con marcatura iniziale $\boldsymbol{x}_0 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$, si assuma che tutte le transizioni abilitate e non in conflitto scattino simultaneamente, mentre in caso di conflitto tra due o più transizioni abilitate ne scatti solo una in maniera casuale.

- Costruire l'albero di raggiungibilità e discutere le proprietà di limitatezza e vivezza della rete o l'eventuale presenza di situazioni di deadlock.
- Determinare tutti gli invarianti della rete e studiare le proprietà di reversibilità e di conservatività.

A valle di queste analisi, si risolvano indipendentemente i seguenti due problemi.

- Progettare un supervisore basato sull'introduzione di posti di controllo (*monitor*) che garantisca l'assenza di deadlock e la vivezza di tutte le transizioni della rete.
- Modificare la marcatura iniziale della rete originaria in una nuova $\boldsymbol{x}'_0$ e pesare opportunamente gli archi in modo da rendere la nuova rete di Petri certamente viva, conservativa e reversibile.

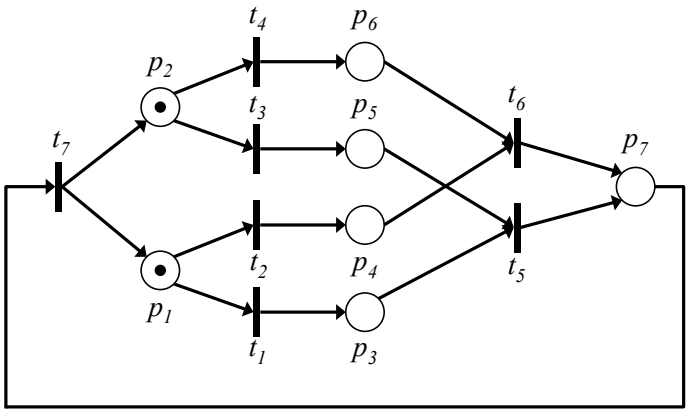

Figura 1: Rete di Petri posti-transizioni con marcatura iniziale $\boldsymbol{x}_0$

[180 minuti; libri aperti]

Soluzioni

16 Gennaio 2015

Esercizio 1

La verifica sulla condizione necessaria si effettua calcolando il fattore di utilizzazione dei task periodici hard real time:

$$U = \frac{2}{12} + \frac{1}{3} + \frac{2}{5} = \frac{54}{60} = 0.9 < 1.$$

Controlliamo se esiste almeno una condizione sufficiente:

$$U_{lsm}(RMPO) = n\left(2^{1/n} - 1\right) = 3\left(2^{1/3} - 1\right) \simeq 0.78.$$

Dato che $U > U_{lsm}$, questa condizione sufficiente non è verificata. Inoltre i tre task non sono legati tra loro da relazioni armoniche. Pertanto non possiamo stabilire a priori se RMPO sia soluzione del problema.

La soluzione dello scheduling RMPO con servizio in background di tipo FIFO è riportata in Fig. 2. Da questa si evince che RMPO è in grado di eseguire lo scheduling dei task hard real time, mentre il task soft real time non può essere portato a termine entro la deadline assoluta, in quanto solo 2 t.u. su 3 vengono eseguite dal μP.

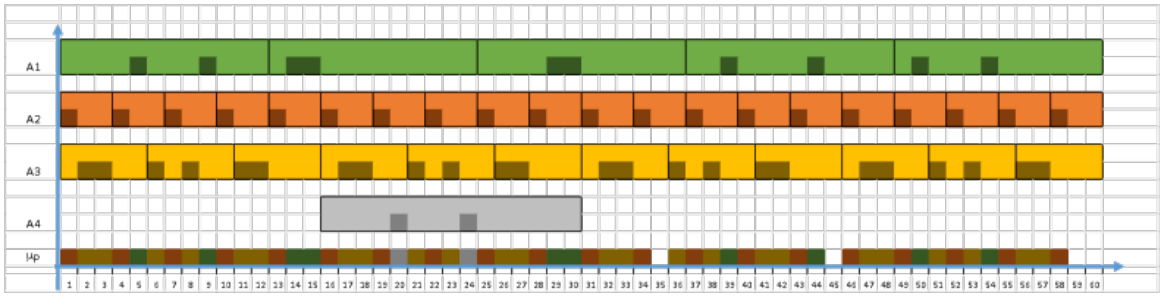

Figura 2: Scheduling RMPO con servizio in background FIFO

La distanza minima tra 3 t.u. liberi del μP è 16 t.u., quindi il task soft real time non può essere eseguito mai entro la deadline. Si può inoltre verificare che modificando uno qualsiasi dei computation time dei task A1, A2 o A3, l'algoritmo RMPO è ancora in grado di schedulare i task periodici hard real time. Quindi il μP, rispetto al problema dato e all'algoritmo di RMPO, *non è* completamente utilizzato.

Esercizio 2

La Fig. 3 mostra una possibile rete di Petri compatta (con il minimo numero di posti e transizioni) che descrive già a sufficienza il comportamento desiderato. Non è strettamente necessario far ricorso a temporizzazione di transizioni o posti. L'interpretazione, che va sempre fornita, di posti (stati) e transizioni (eventi) è riportata nella figura stessa.

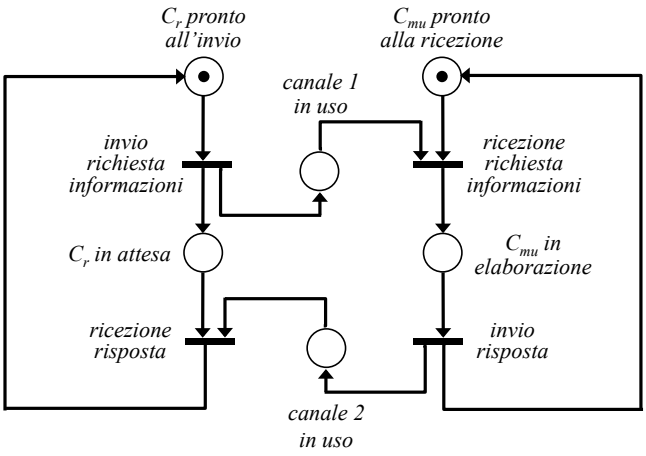

Figura 3: Rete di Petri che modellizza il protocollo di comunicazione tra i controllori C_r e C_{mu}

Esercizio 3

L'albero delle marcature raggiungibili della rete di Petri di Fig. 1 è mostrato in Fig. 4. L'albero è stato costruito in modalità depth-first, espandendo le transizioni abilitate in una data marcatura in ordine lessicografico (t_i prima di t_j, se $i < j$). Si noti che nella costruzione dell'albero di raggiungibilità si fa sempre riferimento a scatti *atomici* delle transizioni, nel senso che non esistono transizioni che scattano simultaneamente.

Il numero di nodi dell'albero è finito e quindi la rete *è limitata*. Analizzando i nodi foglia dell'albero si vede che la rete può reinizializzarsi per alcune sequenze ammissibili di scatto, mentre per altre va in blocco. Quindi la rete *non è viva*.

La matrice di incidenza $\boldsymbol{C}$, di dimensioni (7×7), della rete di Petri di Fig. 1 è la seguente:

$$\boldsymbol{C} = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{pmatrix}. \tag{1}$$

E' immediato verificare (ad esempio con la funzione `rank` in Matlab, ma non solo!) che la matrice $\boldsymbol{C}$ ha rango pari a 5. Essendo la matrice quadrata, la dimensione dello spazio nullo sia di $\boldsymbol{C}$ sia di $\boldsymbol{C}^T$ è pari a $7 - 5 = 2$.

L' equazione che definisce i $T-$invarianti,

$$\boldsymbol{C}\,\boldsymbol{\eta} = \boldsymbol{0},$$

ammette quindi ∞^2 soluzioni linearmente indipendenti. In particolare, i due $T-$invarianti indipendenti con elementi in $\mathbb{Z}^+$ (interi non negativi)

$$\boldsymbol{\eta}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}^T, \qquad \boldsymbol{\eta}_2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}^T$$

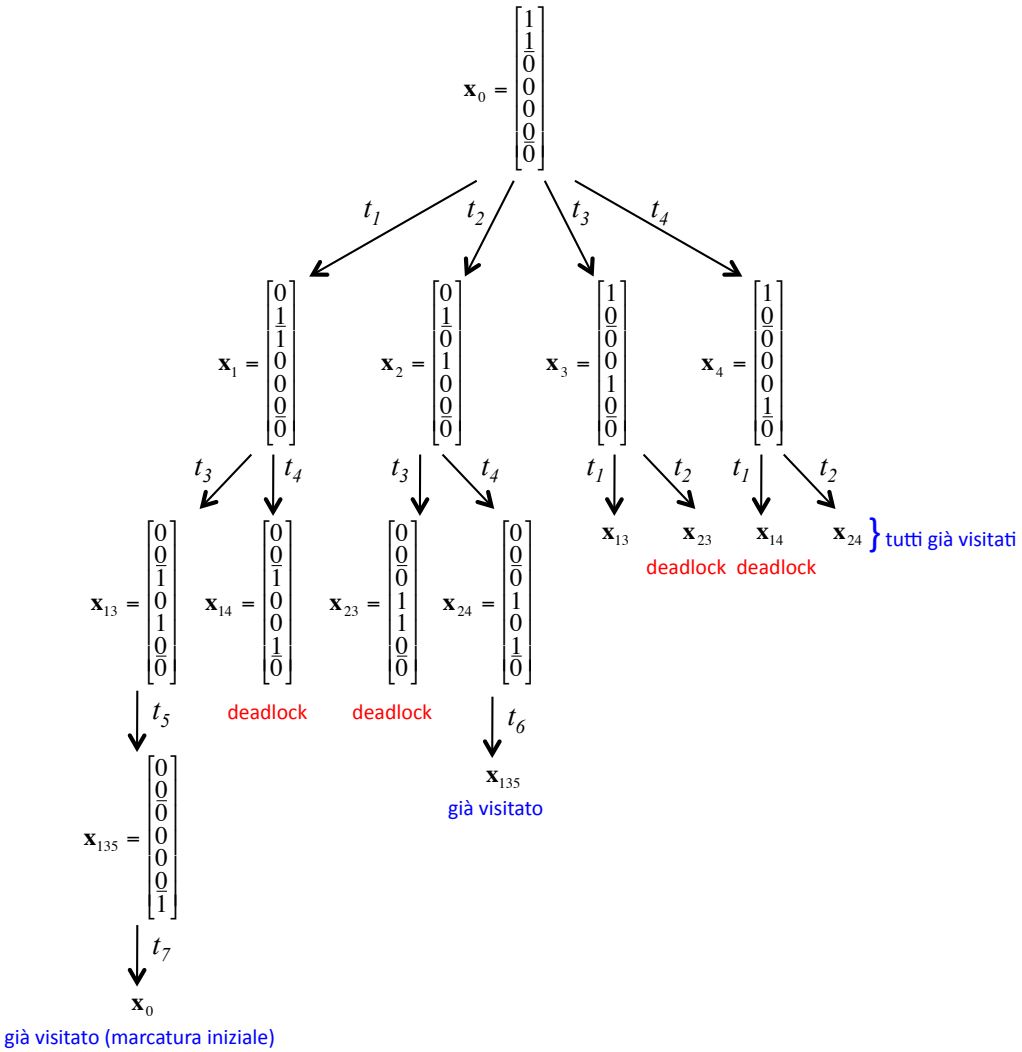

Figura 4: Albero di raggiungibilità della rete di Petri di Fig. 1

sono associati rispettivamente alla sequenza di scatto $\{t_1, t_3, t_5, t_7\}$ (ovvero alla $\{t_3, t_1, t_5, t_7\}$) e alla sequenza di scatto $\{t_2, t_4, t_6, t_7\}$ (ovvero a $\{t_4, t_2, t_6, t_7\}$). Pur essendo quindi la condizione necessaria di reversibilità soddisfatta, queste sono le uniche sequenze di scatto che restituiscono la marcatura iniziale. Come mostrato dall'analisi dell'albero di raggiungibilità, non è vero che $\boldsymbol{x}_0$ sia raggiungibile a partire da qualsiasi marcatura raggiungibile della rete (in particolare la rete si blocca nelle marcature raggiungibili $\boldsymbol{x}_{14}$ e $\boldsymbol{x}_{23}$). Quindi la rete *non è reversibile*.

Anche l'equazione che definisce i $P-$invarianti,

$$\boldsymbol{\gamma}^T \boldsymbol{C} = \mathbf{0}^T \quad \Rightarrow \quad \boldsymbol{C}^T \boldsymbol{\gamma} = \mathbf{0},$$

ammette ∞^2 soluzioni linearmente indipendenti. In particolare, i due $P-$invarianti indipendenti con elementi in $\mathbb{Z}^+$ (e canonici)

$$\boldsymbol{\gamma}_1^T = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, \qquad \boldsymbol{\gamma}_2^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

sono associati rispettivamente alla conservazione del numero di token (pari a $\boldsymbol{\gamma}_i^T \boldsymbol{x}_0 = 1$ per $i = 1, 2$) nel ciclo $\{p_1, p_3 | p_4, p_7\}$ (il simbolo | indica qui un'alternativa) e nel ciclo $\{p_2, p_5 | p_6, p_7\}$ durante l'evoluzione della rete. Poichè ogni posto della rete appartiene al supporto di almeno un $P-$invariante non negativo, la rete *è conservativa* (e quindi anche limitata).

Dall'albero di raggiungibilità della Fig. 4, si osserva che la rete va in deadlock quando scattano in sequenza $\{t_1, t_4\}$ (o $\{t_4, t_1\}$) oppure $\{t_2, t_3\}$ (o $\{t_3, t_2\}$). Nei due casi la rete si blocca con due token presenti nei posti p_3 e p_6 o, rispettivamente, nei posti p_4 e p_5. Per il progetto di un supervisore che eviti queste situazioni, si possono allora imporre le seguenti condizioni operative sui token nelle marcature raggiungibili della rete originale:

$$x(p_3) + x(p_6) \leq 1, \qquad x(p_4) + x(p_5) \leq 1.$$

Questi sono vincoli lineari nella forma

$$\boldsymbol{h}_1^T \boldsymbol{x} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \boldsymbol{x} \leq 1 = k_1, \quad \boldsymbol{h}_2^T \boldsymbol{x} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \boldsymbol{x} \leq 1 = k_2,$$

che devono essere soddisfatti $\forall \boldsymbol{x} \in \mathcal{R}(\boldsymbol{x}_0)$ (l'insieme delle marcature raggiungibili della rete).

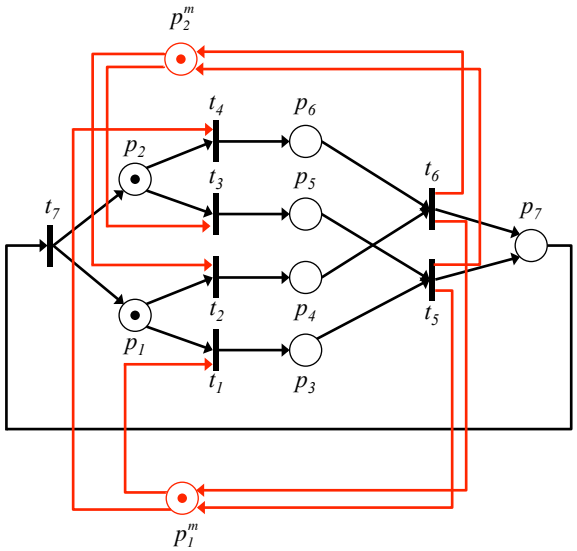

Figura 5: Controllo supervisionato della rete di Petri di Fig. 1 ottenuto con l'aggiunta dei due posti monitor p_1^m e p_2^m

La sintesi del controllore supervisivo (vedi Fig. 5) è fatta considerando due posti monitor p_1^m e p_2^m a cui sono associate due nuove righe nella matrice estesa $\boldsymbol{C}'$ di incidenza

$$\boldsymbol{C}' = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{c}_1^m \\ \boldsymbol{c}_2^m \end{pmatrix}, \qquad \begin{aligned} \boldsymbol{c}_1^m &= -\boldsymbol{h}_1^T \boldsymbol{C} = \begin{pmatrix} -1 & 0 & 0 & -1 & 1 & 1 & 0 \end{pmatrix} \\ \boldsymbol{c}_2^m &= -\boldsymbol{h}_2^T \boldsymbol{C} = \begin{pmatrix} 0 & -1 & -1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{aligned}$$

con marcature iniziali

$$x_0(p_1^m) = k_1 - \boldsymbol{h}_1^T \boldsymbol{x}_0 = k_1 = 1, \qquad x_0(p_2^m) = k_2 - \boldsymbol{h}_2^T \boldsymbol{x}_0 = k_2 = 1.$$

E' facile verificare che la presenza dei posti monitor nel processo controllato porta ad una rete di Petri senza deadlock, viva e reversibile.

Nelle Figg. 6–7 vengono proposte due altre soluzioni al problema del progetto di un supervisore con posti di controllo, ottenute per ispezione e non con un algoritmo esplicito. In entrambi i casi le transizioni risultano tutte vive, ma si forzano a priori delle precedenze nello scatto delle transizioni. La soluzione in Fig. 6 utilizza ancora due soli posti monitor, senza token iniziali: lo scatto (casuale) delle transizioni t_1 o t_2 avviene *sempre prima* di uno delle due transizioni t_3 (abilitata solo se è scattata precedentemente t_1) o t_4 (abilitata solo se è scattata precedentemente t_2). La soluzione in Fig. 7 usa invece quattro posti monitor, con un token iniziale in p_1^m; viene imposto un ordine ancora più stretto nello scatto delle transizioni, con l'unica sequenza di scatto abilitata $\{t_1, t_3, t_5, t_7, t_4, t_2, t_6, t_7\}$ che reinizializza la rete (non ci sono più transizioni in conflitto).

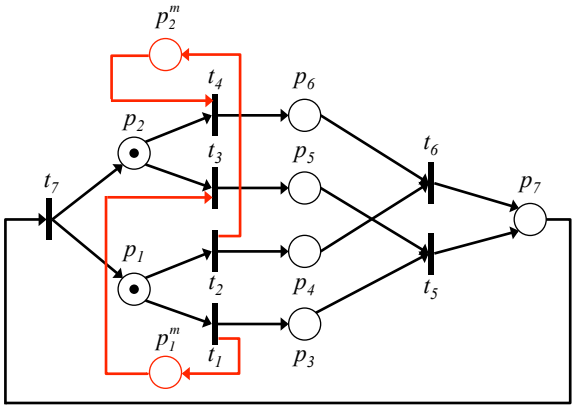

Figura 6: Prima alternativa al supervisore di Fig. 5, ancora con l'aggiunta di due posti monitor

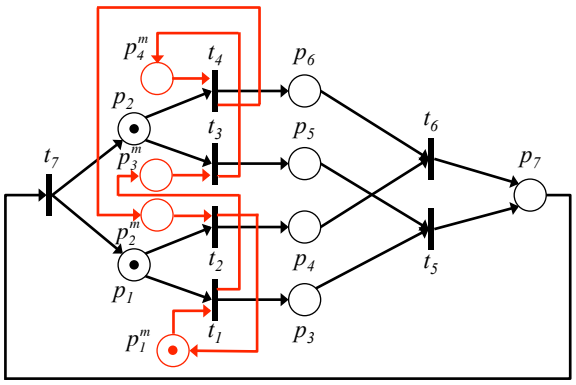

Figura 7: Seconda alternativa al supervisore di Fig. 5, con l'aggiunta di quattro posti monitor

Il secondo problema posto è risolto ad esempio dalla rete in Fig. 8. Si noti che tutte le transizioni hanno ora il *fan-in* uguale al *fan-out*, ossia il numero totale di token prelevati allo scatto dai posti in ingresso è uguale a quello totale depositato nei posti uscita. Pertanto la rete è certamente conservativa, e quindi anche limitata. Inoltre, ci sono risorse sufficienti per non creare conflitti tra transizioni abilitate contemporaneamente. Con la logica di scatto indicata nel testo, l'unica

evoluzione possibile per questa rete (indicando con $\|$ le transizioni che scattano in parallelo) è

$$\boldsymbol{x}'_0 = \begin{pmatrix} 2 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \{t_1 \| t_2 \| t_3 \| t_4\} \rightarrow \boldsymbol{x}'_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \rightarrow \{t_5 \| t_6\} \rightarrow \boldsymbol{x}'_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 4 \end{pmatrix} \rightarrow \{t_7\} \rightarrow \boldsymbol{x}'_0.$$

La rete è quindi reversibile. Inoltre la rete è viva, perché tutte le transizioni appaiono in questa sequenza ciclica e sono abilitate allo scatto.

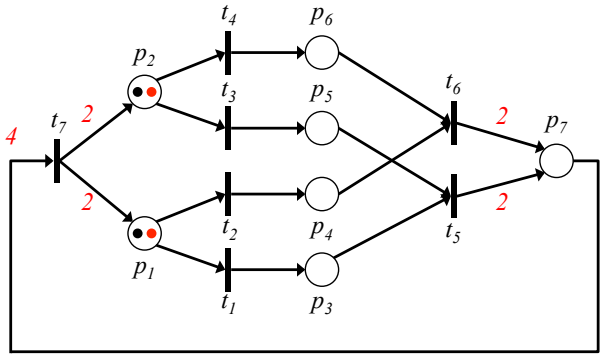

Figura 8: Modifica della rete di Petri di Fig. 1 con l'aggiunta di token nella marcatura iniziale $\boldsymbol{x}'_0$ e di pesi non unitari (in rosso) su alcuni archi

Un'altra rete che soddisfa gli stessi requisiti del problema è mostrata in Fig. 9. Rispetto alla precedente soluzione, è ottenuta facendo meno modifiche alla rete originaria.

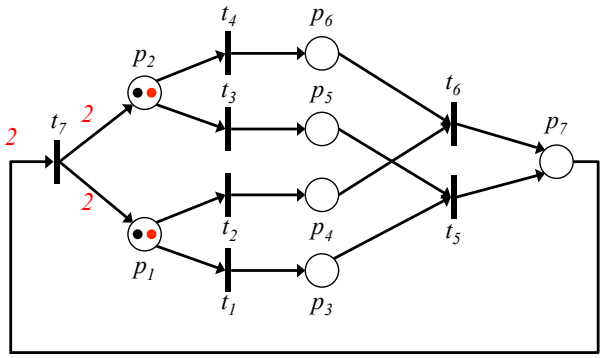

Figura 9: Una soluzione alternativa alla rete di Fig. 8

* * * * *