Controllo decentralizzato di reti di Petri mediante posti monitor

Maria Paola Ciullo

Tesi di Laurea Specialistica in Ingegneria Elettronica

Corso di LS in Ingegneria Elettronica, Università degli Studi di Cagliari,

Dipartimento di Ing. Elettrica ed Elettronica

Relatori: Alessandro Giua, Carla Seatzu

15 Luglio 2009

Ringraziamenti

Ringrazio il Professor Alessandro Giua e la Professoressa Carla Seatzu che mi hanno permesso di svolgere questa tesi e mi hanno seguito con grande disponibilità nella realizzazione di questo lavoro.

II

Sommario

In questa tesi si considera il problema della determinazione di un insieme di posti monitor decentralizzati per reti posto/transizione. Tale struttura di controllo deve rispettare una specifica globale sul comportamento della rete data, espressa mediante specifiche di mutua esclusione generalizzata (GMEC), che determinano l'insieme di marcature legali della rete.

La tesi si basa su due approcci alla risoluzione di tale problema, recentemente apparsi in letteratura. Il primo, basato sulla programmazione non lineare intera, porta a soluzioni ottimali solo con specifiche assunzioni sul peso dei coefficienti delle GMEC. Il secondo rilassa i vincoli imposti precedentemente e garantisce massimalità e buona fairness tra i posti.

L'implementazione di tali algoritmi, denominati HB (Hypercube Bound) e HMB (Hypercube Maximal Bound), mirati a ricercare i boundaries di un ipercubo intero interno a un insieme di GMEC, rappresenta il fulcro di tale lavoro. La funzionalità del software sviluppato con Matlab è stata testata con numerosi esempi, e in alcuni casi si è quantitativamente misurata la bontà delle soluzioni ottenute valutando il numero di marcature presenti all'interno dell'ipercubo intero determinato con HMB.

Abstract

In this thesis the problem of determining a set of decentralized monitors for place/transition nets is considered. This control structure has to respecte a global specification on the behaviour of the given net, expressed by generalized mutual exclusion constraints (GMEC), that determine the net's legal markings set.

The thesis is based on two approaches to risolution of this problem recently appeared in literature. The first one, based on integer non linear programming, leads to optimal solutions only with specific assumptions on the weights of GMEC. The second one relaxes the previously imposed constraints and guarantees maximallity and good fairness among places.

The implementation of these alghoritms, called HB (Hypercube Bound) and HMB (Hypercube Maximal Bound), which search for the boundaries of the integer inner hypercube in the GMEC, is the core of this work. The functionality of the software developed with Matlab has been tested with a several examples, and in some cases the accuracy of computed solutions has been misured in a quantitative way, evaluating the number of the markings in the interior of the integer hypercube determined with HMB.

Indice

Capitolo 1

Introduzione

1.1 Stato dell'arte

Nel settore dell'automatica, il concetto di decentralizzazione è particolarmente importante perchè consente a un sistema che descrive un processo di avere una diversa struttura non basata su un unico controllore. Ciò comporta notevoli vantaggi visto che una struttura di tipo centralizzato non è implementabile per sistemi di grandi dimensioni, così si propone una soluzione decentralizzata in grado di far lavorare indipendentemente ogni controllore garantendo quindi maggiore robustezza ai guasti.

Un controllore decentralizzato consiste di diversi supervisori locali, ognuno dei quali controlla e osserva una parte del sistema per cui essi possono essere utilizzati in diverse applicazioni che riguardano la produzione, la ricerca di guasti, i protocolli di comunicazione, ecc.

Lo studio del controllo decentralizzato è sempre stato di grande interesse per quel che riguarda i sistemi ad eventi discreti (SED) mediante approcci con linguaggi formali utilizzando automi; al contrario, lo stesso problema risolto usando le reti di Petri non ha ricevuto grande attenzione nonostante la loro rappresentazione compatta costituisca un aiuto per ridurre la complessità dei problemi di supervisori di controllo decentralizzati.

Il problema del controllo decentralizzato risolto con l'ausilio degli automi è stato considerato in due diverse versioni: con comunicazione tra i controllori o senza comunicazione e sono stati proposti differenti approcci per la realizzazione della

soluzione.

Il problema è stato ampiamente trattato in [7] in cui vengono discusse le condizioni necessarie e sufficienti per l'esistenza di una soluzione, le quali richiedono che il comportamento del sistema a ciclo chiuso sia descritto da un linguaggio regolare. L'insieme degli eventi viene diviso in un sottoinsieme di eventi controllabili e un sottoinsieme di eventi non controllabili. Il controllore osserva una sequenza di eventi e abilita o disabilita alcuni eventi, poichè esso può prendere decisioni solo sulla base di ciò che osserva. Si distingue tra supervisori locali (Local Problem) i quali, come detto, possono controllare solo alcuni sottoinsiemi di eventi controllabili, e supervisori globali (Global Problem) che possono agire sull'intero insieme di eventi.

Per quel che riguarda l'utilizzo delle reti di Petri il problema non è stato ancora ampiamente trattato. In particolare, tra i metodi proposti in letteratura vi è il *supervision based on place invariants* (SBPI) [15] che rappresenta una tecnica efficiente per la realizzazione di un supervisore decentralizzato che impone alla rete il rispetto di un insieme di condizioni date in termini di specifiche di mutua esclusione generalizzate (GMEC). Tale tecnica è stata usata con successo nell'ambiente centralizzato delle reti di Petri e in [5] si è pensato di estenderla all'ambiente decentralizzato.

Questa struttura classifica le specifiche in ammissibili e inammissibili, in particolare è possibile trasformare queste ultime in ammissibili e renderle più restrittive per fare in modo che poi possano essere rispettate. Questa trasformazione è necessaria poichè una specifica ammissibile è l'equivale di una specifica controllabile e osservabile. Inoltre viene introdotta la condizione di d-ammissibilità, estensione dell'ammissibilità all'ambiente decentralizzato, la quale individua i vincoli per i quali può essere realizzato facilmente un supervisore. Se un vincolo è non d-ammissibile in [6] si fornisce un approccio algoritmico che permetta al vincolo d-ammissibile di abilitare la comunicazione tra eventi. Quindi in questo nuovo insieme ammissibile si individuano i vincoli che permettono di eleborare facilmente un supervisore.

In [5] è stato proposto di trovare dei supervisori locali in cui ciascuno osservi e controlli una parte del sistema rispettando una specifica globale con delle tecniche di programmazione lineare intera. Esse ricercano i valori di alcune incognite il cui ruolo è quello di trasformare delle specifiche globali (multiple sets) in specifiche locali (single set) che possono essere implementate dai suddetti supervisori locali. La dimensione del problema dipende dalla dimensione della rete di Petri e potrebbe non essere finita.

1.2 Motivazioni e obiettivi della tesi

Di recente è apparso in lettaratura un altro approccio [1] che vuole studiare il problema della determinazione di un insieme di posti monitor decentralizzati per reti posto/transizione rispettando una specifica globale sul comportamento della rete data in termini di specifiche di mutua esclusione generalizzata $(\boldsymbol{W}, \boldsymbol{k})$ dove $\boldsymbol{W} = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, ...\boldsymbol{w}_{n_c}^T]^T \in \mathbb{Z}^{n_c \times m}$ e $\boldsymbol{k} = [k_1, k_2, ...k_{n_c}]^T \in \mathbb{Z}^{n_c}$ e l'insieme di marcature legali è definito da $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{W} \cdot \boldsymbol{m} \leq \boldsymbol{k}\}$.

Questa tesi considera questo approccio, studiandone le soluzioni e implementandone gli algoritmi proposti.

In [1] vengono considerate due diverse formulazioni del problema.

Il primo caso, *caso A*, considera come principale obiettivo quello di massimizzare la cardinalità dell'insieme di marcature legali per il controllo decentralizzato. Questo problema risulta essere piuttosto complesso e non è possibile trovare una soluzione se vengono fatte delle assunzioni generali:

- i pesi delle GMEC possono assumere qualsiasi valore (positivo o negativo);
- i supporti delle GMEC decentralizzate vengono scelti arbitrariamente.

La soluzione presentata in [1] per un problema con le suddette condizioni si basa sulla programmazione matematica e garantisce l'ottimalità della soluzione solo nel caso in cui i pesi della GMEC globale siano positivi e l'insieme decentralizzato dei posti P_i sia singleton.

Per semplificare il problema viene proposto un secondo approccio, *caso B*, il cui obiettivo è quello di determinare un insieme di marcature legali decentralizzate massimale; infatti per alcuni insiemi decentralizzati non esiste una direzione in cui tale insieme possa essere aumentato senza violare la specifica globale. In particolare tra tutti gli insiemi massimali si seleziona quello che garantisce una buona fairness tra i posti.

In questo secondo caso non vengono fatte assunzioni sui pesi dei coefficienti delle GMEC globali. La soluzione a tale problema è proposta in [1] e si basa su 2 passi.

- Il primo passo consiste nel calcolare una soluzione per il problema di controllo sotto l'assunzione che i supporti di tutte le GMEC siano singleton. Si

determina un punto interno $\boldsymbol{c} \in \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ e si definisce tale punto $\boldsymbol{c}$ centro di un nuovo sistema di coordinate.

- Il secondo passo si basa sulla valutazione e sulla massimizzazione di un box intero interno $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{l} \leq \boldsymbol{m} \leq \boldsymbol{u}\}$ incluso nell'insieme di marcature legali definite dalla GMEC globale $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

La soluzione a tale problema si basa su un algoritmo iterativo. In particolare in questa tesi vengono studiati due diversi algoritmi che consentono di trovare un box intero interno che sia anche massimale e che, in alcuni casi, possono portare allo stesso risultato, nel caso il primo algoritmo determini l'inner box massimo.

1. Il primo algoritmo consente di determinare un inner box.
2. Il secondo algoritmo massimizzerà l'inner box determinato col primo algoritmo.

L'implementazione dei due algoritmi è stata eseguita mediante l'utilizzo del software Matlab che ha permesso di determinare i boundaries di tale box.

1.3 Organizzazione della tesi

Nel capitolo 2 verrà descritto il formalismo relativo alle reti di Petri e alle GMEC e verrà fatto un breve cenno ai sistemi ad eventi discreti per cercare di contestualizzare gli argomenti esposti. Dopo tale richiamo il capitolo si dividerà in due parti. Nella prima parte si esporrano le nozioni principali sulle reti di Petri, in particolare si vedrà la struttura di una rete di Petri, definita per mezzo di una struttura del tipo $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ e verrà spiegato come scrivere una matrice di incidenza associata a una data struttura. Si esporrà l'importante concetto di marcatura, che consente di definire lo stato di una rete di Petri e, a proposito di tale concetto verrà introdotta la definizione di abilitazione e di scatto. Nella seconda parte del capitolo sarà definito il concetto di specifica statica detta di mutua esclusione generalizzata (GMEC). Verrà introdotto il concetto di posto monitor come struttura di supervisione focalizzandosi sulla realizzazione di tale struttura e su alcune caratteristiche che verranno utilizzate in questa tesi.

Nel capitolo 3, dopo avere esposto le principali definizioni geometriche riguardanti il problema decentralizzato, si procederà a una sua esposizione dettagliata.

In particolare si distinguerà una possibile soluzione basata sulla programmazione matematica e una possibile soluzione senza assunzioni generali. Quest'ultima si differenzia ulteriormente in due casi: con posti singleton e non singleton. Sarà il primo caso quello su cui verrà focalizzata l'attenzione e che verrà risolto per mezzo di due step che condurrano alla formulazione di algoritmi iterativi.

Nel capitolo 4 si vedrà come è stato risolto il problema di programmazione lineare intero per la ricerca del centro $\boldsymbol{c}$ e del lato 2τ dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$; si farà uso per la risoluzione di tale problema, di una libreria di Matlab chiamata *glpkmex.lib*, che consente di risolvere problemi di programmazione lineare. Successivamente verranno analizzati e sviluppati i principali passi eseguiti per l'implementazione dei due algoritmi proposti per la ricerca dei boundaries dell'ipercubo massimale; in particolare, le funzioni create verranno analizzate al fine di descrivere i parametri utilizzati nell'implementazione degli algoritmi. Verrà esposto il significato di ciascuna funzione creata specificando quali sono gli argomenti definiti in ingresso, in uscita e la funzionalità implementata. Per lo sviluppo di questo programma si è fatto uso del software Matlab. Prima di concludere il capitolo verrà esposta la soluzione proposta per risolvere un problema che conta il numero di marcature presenti all'interno dell'ipercubo intero massimale. Questo aspetto risulta essere particolarmente innovativo rispetto a quanto è presente in letteratura sull'argomento trattato in questa tesi.

Nel capitolo 5 l'attenzione si focalizzerà su alcuni esempi che verranno risolti per mezzo degli algoritmi implementati e di cui verranno presentati i risultati. Ogni esempio avrà caratteristiche differenti poichè l'obiettivo è quello di dimostrare l'applicabilità degli algoritmi a qualsiasi tipo di GMEC senza alcuna limitazione. Gli esempi presi in considerazione verranno eseguiti in tutti i loro step dal punto di vista analitico e verranno presentati i risultati più importanti al fine della determinazione dei boundaries. In particolare si propongono cinque esercizi, due dei quali possono considerarsi degli esempi didattici poichè non hanno alcun significato fisico e non si conosce la rete di Petri che fa riferimento all'insieme di GMEC, due invece derivano dalla modellazione di una rete di Petri e rappresentano un sistema fisico in grado di descrivere un processo di workflow e infine uno mirato alla valutazione dei tempi di elaborazione.

Il capitolo 6 è il capitolo conclusivo in cui verranno formulate le conclusioni scaturite da questo lavoro di tesi e si analizzeranno i suoi contributi per eventuali sviluppi futuri.

Questa tesi apporta dei contributi differenti alla ricerca.

- Lo sviluppo di un software in Matlab che permette di eseguire gli algoritmi trattati, con alcuni miglioramenti, in maniera agevole anche in presenza di un elevato numero di posti.
- Una modifica del campo di definizione delle variabili del problema di programmazione lineare fondamentale per lo sviluppo del software.
- La realizzazione di una funzione in grado di valutare la bontà della soluzione in termini di cardinalità dello spazio di stato.
- Una serie di simulazioni che dimostrano la validità dell'approccio presentato.

Capitolo 2

Reti di Petri e GMEC

2.1 Introduzione

In questo capitolo viene introdotto il formalismo legato alle reti di Petri e alle specifiche di mutua esclusione generalizzate (GMEC). Per contestualizzare tali argomenti viene fatto un breve cenno ai sistemi ad eventi discreti.

Un sistema ad eventi discreti (SED) è un sistema il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi che individuano lo svolgimento di attività di durata non necessariamente nota. Esso è formalmente caratterizzato da:

- un insieme E degli eventi accadibili;
- uno spazio di stato costituito da un insieme discreto X;
- un'evoluzione dello stato regolata dagli eventi: lo stato evolve nel tempo solo in dipendenza dell'accadimento di eventi asincroni appartenenti all'insieme E.

L'equazione che descrive l'evoluzione dello stato a partire dallo stato iniziale x_0 è

$$x_{k+1} = \delta(x_k, e_k)$$

dove

- x_{k+1} è lo stato del sistema dopo l'accadimento dell'evento k-esimo;
- e_k è il k-esimo evento accaduto dall'istante iniziale considerato, che fa transire lo stato da x_k a $x_k + 1$;
- $\delta : X \times E \longrightarrow X$ è la funzione di transizione di stato.

I sistemi ad eventi discreti si dividono in due grandi famiglie: quella dei modelli logici e quella dei modelli temporizzati, a seconda del tipo di traccia degli eventi adottata.

Nei modelli logici la traccia degli eventi è costituita da una sequenza di eventi $\{e_1, e_2, ...\}$ in ordine di occorrenza, senza avere alcuna informazione sui tempi di occorrenza degli eventi. Dato uno stato iniziale x_0, la traiettoria dello stato verrà costruita nel tempo come la sequenza di stati $\{x_0, x_1, x_2...\}$ risultanti dall'accadimento della sequenza di eventi, ma non è possibile specificare gli istanti di tempo in cui avvengono le transizioni. Tali modelli rendono agevole lo studio delle proprietà qualitative del sistema e consentono di effettuare l'analisi strutturale di un SED.

Nei modelli temporizzati la traccia degli eventi è costituita da una sequenza di coppie $\{(e_1, \tau_1), (e_2, \tau_2), ...\}$ dove ogni evento e_i è accoppiato al suo tempo di accadimento τ_i, eventualmente stocastico. Dato uno stato iniziale x_0, la traiettoria dello stato sarà ancora la sequenza di stati $\{x_0, x_1, x_2...\}$ risultanti dall'accadimento della sequenza di eventi ma, in questo caso si sa che le transizioni di stato avvengono negli istanti di occorrenza degli eventi. Questi modelli sono utili qualora si voglia effettuare l'analisi prestazionale di un SED poichè permettono di studiare i diversi comportamenti del sistema nel tempo.

2.2 Reti di Petri

Le reti di Petri sono un modello di sistema a aventi discreti [2] che trae origine dal lavoro di Carl Adam Petri, un ricercatore tedesco, che nel 1962 discusse la sua tesi di dottorato dal titolo *"kommunication mit Automaten"* in cui presentò un nuovo modello logico che in seguito prese il suo nome.

Le reti di Petri si dividono in logiche e temporizzate; quest'ultime si dividono in deterministiche e stocastiche.

In questa tesi si tratteranno le reti di Petri logiche considerandole come un modello logico che non consente di rappresentare la temporizzazione degli eventi, ma solo l'ordine con cui questi si verificano.

Fra i vari modelli a eventi discreti, le reti di Petri hanno una importanza predominante a causa di vari fattori:

- forniscono un formalismo grafico e matematico per la modellazione dei SED;
- permettono di dare una rappresentazione compatta di sistemi con un grande spazio di stati;
- permettono di rappresentare il concetto di concorrenza, cioè di attività che possono venire svolte parallelamente;
- consentono una rappresentazione modulare; cioè se un sistema è composto da più sottosistemi che interagiscono fra di loro, è generalmente possibile rappresentare ciascun sottosistema come una semplice rete e poi, mediante operatori di rete, unire le varie sottoreti per ottenere il modello del sistema complessivo.

2.2.1 Struttura delle reti posto/transizione

Una rete di Petri P/T è un grafo bipartito, orientato e pesato. I due tipi di vertici sono detti posti, e vengono rappresentati da cerchi, e transizioni e vengono rappresentati da barre. Gli archi, devono essere orientati e permettono di collegare i posti alle transizioni e viceversa.

Definizione 2.2.1. *Una* rete posto/transizione *o* rete P/T *è una struttura* $N = (P, T, \mathbf{Pre}, \mathbf{Post})$ *dove:*

- $P = \{p_1, p_2, ..., p_m\}$ *è l'insieme degli* m posti*;*
- $T = \{t_1, t_2, ..., t_n\}$ *è l'insieme delle* n transizioni*;*
- $\mathbf{Pre} : P \times T \longrightarrow N$*: è la* funzione di pre-incidenza *che specifica gli archi diretti dai posti alle transizioni (detti archi "pre") e viene rappresentata mediante una matrice* $m \times n$*;*

- $\mathbf{Post} : P \times T \longrightarrow N$*: è* funzione di post-incidenza *che specifica gli archi diretti dalle transizioni ai posti (detti archi "post") e viene rappresentata mediante una matrice* $m \times n$*.*

Si suppone che $P \cap T = \emptyset$, cioè posti e transizioni sono insiemi disgiunti e che $P \cup T \neq \emptyset$, cioè la rete è costituita da almeno un posto o da una transizione.

Le matrici $\mathbf{Pre}$ e $\mathbf{Post}$ sono delle matrici di interi non negativi. Si denota con $\mathbf{Pre}(\cdot, t)$ la colonna della matrice Pre relativa alla transizione t, e con $\mathbf{Pre}(p, \cdot)$ la riga della matrice $\mathbf{Pre}$ relativa al posto p. La stessa notazione vale per la matrice $\mathbf{Post}$. L'informazione sulla struttura di rete contenuta nelle matrici $\mathbf{Pre}$ e $\mathbf{Post}$ può essere compattata in un'unica matrice, detta matrice di incidenza.

Definizione 2.2.2. *Data una rete* $N = (P, T, \mathbf{Pre}, \mathbf{Post})$*, con* m *posti ed* n *transizioni, la* matrice di incidenza $C : P \times T \longrightarrow Z$ *è la matrice* $m \times n$ *definita come:*

$$\boldsymbol{C} = \mathbf{Post} - \mathbf{Pre}$$

il cui generico elemento di $\boldsymbol{C}$ *vale* $C(p, t) = Post(p, t) - Pre(p, t)$*.*

Un elemento negativo è associato a un arco *pre* dal posto alla transizione, mentre un elemento positivo è associato a un arco *post* diretto dalla transizione al posto.

Nel compattare le matrice $\mathbf{Pre}$ e $\mathbf{Post}$ la matrice di incidenza $\boldsymbol{C}$ perde informazioni sulla struttura della rete e non permette di poterla ricostruire.

Esempio 2.2.1. *In Figura (2.1) è rappresentata la rete* $N = (P, T, \mathbf{Pre}, \mathbf{Post})$ *con insieme di posti* $P = \{p_1, p_2, p_3, p_4\}$ *e insieme delle transizioni* $T = \{t_1, t_2, t_3, t_4, t_5\}$*, le matrici Pre e Post valgono:*

$$\mathbf{Pre} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{Post} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

La matrice di incidenza vale:

$$\boldsymbol{C} = \begin{pmatrix} 0 & -1 & 0 & 0 & 1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}.$$

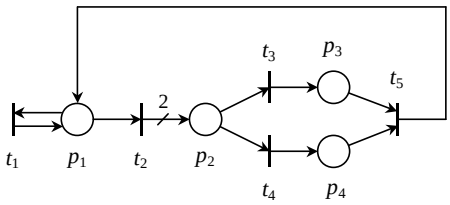

Figura 2.1: Rete di Petri posto-transizione

Si noti che $Post(p_2, t_2) = 2$ e dunque vi sono due archi che vanno dalla transizione t_2 al posto p_2. Nella Figura, invece di rappresentare i due archi è usata una notazione semplificata che consiste nel rappresentare un solo arco avente per etichetta un numero (2 in questo caso) che indica la sua molteplicità.

Infine, data una transizione $t \in T$ si definiscono i seguenti sistemi di posti:

${}^\bullet t = \{p \in P \mid Pre(p, t) > 0\}$: è l'insieme dei posti in ingresso a t.
$t^\bullet = \{p \in P \mid Post(p, t) > 0\}$: è l'insieme dei posti in uscita da t;

dato un posto $p \in P$ si definiscono i seguenti insiemi di transizioni:

${}^\bullet p = \{t \in T \mid Post(p, t) > 0\}$: è l'insieme delle transizioni in ingresso a p.
$p^\bullet = \{t \in T \mid Pre(p, t) > 0\}$: è l'insieme delle transizioni in uscita da p.

Ad esempio nella rete in Figura (2.1) vale ${}^\bullet t_2 = \{p_1\}$, $t_2^\bullet = \{p_2\}$, ${}^\bullet p_2 = \{t_2, t_5\}$, $p_2^\bullet = \{t_3, t_4\}$.

2.2.2 Marcatura e sistema di rete

La *marcatura* è un importante concetto che consente di definire lo stato di una rete P/T.

Definizione 2.2.3. *Una* marcatura *è una funzione $M : P \longrightarrow N$ che assegna a ogni posto un numero intero non negativo di marche (o gettoni).*

Tali marche vengono rappresentate graficamente da pallini all'interno dei posti.

Solitamente si indica una marcatura come un vettore colonna con m componenti, tante quanti sono i posti presenti nella rete. Considerando l'esempio in Figura (2.1), una marcatura possibile $\boldsymbol{M}$ è $M(p_1) = 1, M(p_2) = M(p_3) = M(p_4) = 0$ come mostrato in Figura (2.2).

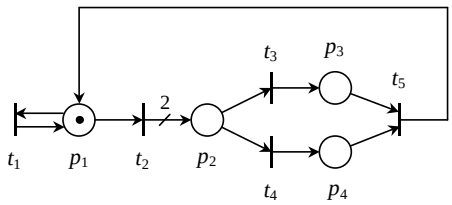

Figura 2.2: Evoluzione di una rete marcata. Marcatura iniziale.

Definizione 2.2.4. *Una rete N con una marcatura iniziale $\boldsymbol{M}_0$ è detta* rete marcata o sistema di rete*, e viene indicata come $\langle N, \boldsymbol{M}_0 \rangle$.*

2.2.3 Abilitazione e scatto

Definizione 2.2.5. *Una* transizione t *è* abilitata *dalla marcatura $\boldsymbol{M}$ se*

$$\boldsymbol{M} \geqslant \mathbf{Pre}(\cdot, t)$$

cioè se ogni posto $p \in P$ della rete contiene un numero di marche pari o superiore a $Pre(p,t)$. Per indicare che t è abilitata da $\boldsymbol{M}$ si scrive $\boldsymbol{M}[t\rangle$. Per indicare che t' non è stata abilitata da $\boldsymbol{M}$ si scrive $\neg\boldsymbol{M}[t'\rangle$.

Osservando una rete marcata si può capire se una transizione t è abilitata; ogni posto in ingresso a una transizione, cioè ogni posto $p \in {}^\bullet t$, deve avere un numero di marche pari o superiore agli archi $''pre''$ che vanno dal posto alla transizione.

Definizione 2.2.6. *Una transizione t abilitata da una marcatura $\boldsymbol{M}$ può* scattare*. Lo scatto di t rimuove $Pre(p,t)$ marche da ogni posto $p \in P$ e aggiunge $Post(p,t)$ marche in ogni posto $p \in P$, determinando una nuova marcatura $\boldsymbol{M}'$. Cioè vale:*

$$\boldsymbol{M}' = \boldsymbol{M} - \mathbf{Pre}(\cdot, t) + \mathbf{Post}(\cdot, t) = \boldsymbol{M} + \boldsymbol{C}(\cdot, t).$$

Per indicare che lo scatto di t da $\boldsymbol{M}$ determina la marcatura $\boldsymbol{M}'$ si scrive $\boldsymbol{M}[t\rangle\boldsymbol{M}'$.

Definizione 2.2.7. *Una sequenza di transizioni $\sigma = t_{j_1} t_{j_2} ... t_{j_r} \in T^*$ è* abilitata *da una marcatura $\boldsymbol{M}$ se: la transizione t_{j_1} è abilitata da $\boldsymbol{M}$ e il suo scatto porta a $\boldsymbol{M}_1 = \boldsymbol{M} + \boldsymbol{C}(\cdot, t_{j_1})$; la transizione t_{j_2} è abilitata da $\boldsymbol{M}_1$ e il suo scatto porta a $\boldsymbol{M}_2 = \boldsymbol{M}_1 + \boldsymbol{C}(\cdot, t_{j_2})$ ecc. Una sequenza abilitata σ viene anche detta* sequenza di scatto *e ad essa corrisponde la traiettoria:*

$$\boldsymbol{M}[t_{j_1}\rangle \boldsymbol{M}_1[t_{j_2}\rangle \boldsymbol{M}_2 ... [t_{j_r}\rangle M_r.$$

Per indicare che la sequenza σ è abilitata da $\boldsymbol{M}$ si scrive $\boldsymbol{M}[\sigma\rangle$. Per indicare che lo scatto di σ da $\boldsymbol{M}$ determina la marcatura $\boldsymbol{M}'$ si scrive $\boldsymbol{M}[\sigma\rangle \boldsymbol{M}'$.

Ad esempio nella rete in Figura (2.2) una possibile sequenza di transizioni abilitata dalla marcatura data è $\sigma = t_1 t_1 t_2 t_3$ il cui scatto porta alla marcatura iniziale $M_0 = [\ 0 \quad 1 \quad 1 \quad 0\ ]^T$.

Definizione 2.2.8. *Il* comportamento (o linguaggio) *di una rete marcata* $\langle N, \boldsymbol{M}_0\rangle$ *è l'insieme delle sequenze di scatto abilitate dalla marcatura iniziale, cioè l'insieme:*

$$L(N, \boldsymbol{M}_0) = \{\sigma \in T^* \mid \boldsymbol{M}_0[\sigma\rangle\}.$$

Definizione 2.2.9. *Una marcatura* $\boldsymbol{M}$ *è detta* raggiungibile *in* $\langle N, \boldsymbol{M}_0\rangle$ *se esiste una sequenza di scatto tale che* $\boldsymbol{M}_0[\sigma\rangle M$. *L'*insieme di raggiungibilità *di una rete marcata* $\langle N, \boldsymbol{M}_0\rangle$ *è l'insieme delle marcature che possono venir raggiunte a partire dalla marcatura iniziale, cioè l'insieme:*

$$R(N, \boldsymbol{M}_0) = \{\boldsymbol{M} \in N^m \mid \exists \sigma \in L(N, \boldsymbol{M}_0) : \boldsymbol{M}_0[\sigma\rangle \boldsymbol{M}\}.$$

La marcatura di una rete ha un duplice significato: da un lato indica lo stato in cui si trova il sistema, dall'altro specifica quali transizioni sono abilitate.

2.2.4 Equazione di stato

Proposizione 2.2.1. *Sia* $\langle N, \boldsymbol{M}_0\rangle$ *una rete marcata e sia* $\boldsymbol{C}$ *la sua matrice di incidenza. Se* $\boldsymbol{M}$ *è raggiungibile da* $\boldsymbol{M}_0$ *scattando la sequenza di transizioni* σ *vale:*

$$\boldsymbol{M} = \boldsymbol{M}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}.$$

$\boldsymbol{\sigma}$ *è un* vettore di scatto *che ha tante componenti quante sono le transizioni.*

2.2.5 Proprietà comportamentali

Le proprietà comportamentali sono quelle proprietà che dipendono sia dalla struttura della rete, sia dalla marcatura iniziale. Verranno esposte di seguito le principali.

Raggiungibilità. La raggiungibilità di una marcatura permette di studiare quali sono i possibili stati in cui un sistema può trovarsi a partire da un dato stato.

Limitatezza. La limitatezza di un posto o di una rete marcata permette di verificare che non vi siano eccedenza e di dimensionare opportunamente le risorse.

Definizione 2.2.10. *Un posto p è detto* k-limitato *in $\langle N, \boldsymbol{M}_0 \rangle$ se per ogni marcatura raggiungibile $\boldsymbol{M}$ vale $M(p) \leq k$. Un posto 1-limitato è detto* sano (o binario). *Una rete marcata $\langle N, \boldsymbol{M}_0 \rangle$ è* k-limitata *se ogni suo posto è* k-limitato. *Una rete 1-limitata è detta* sana (o binaria)

Proposizione 2.2.2. *Una rete marcata $\langle N, \boldsymbol{M}_0 \rangle$ è limitata se e solo se ha un insieme di raggiungibilità finito.*

Dimostrazione. (Se) Sia P l'insieme dei posti della rete. Se l'insieme di raggiungibilità è finito, è possibile calcolare

$$k = max\{M(p) \mid \boldsymbol{M} \in R(N, \boldsymbol{M}_0), p \in P\},$$

e chiaramente la rete è k-limitata.

(Solo se) Se la rete è k-limitata, si definisca il vettore $\boldsymbol{k} = [\ k \quad k \quad k \quad ... \quad k\ ]^T \in \mathbb{N}^m$. Vale $R(N, \boldsymbol{M}_0) \subseteq \{\boldsymbol{M} \in \mathbb{N}^m \mid \boldsymbol{M} \leq k\}$ e quest'ultimo insieme è chiaramente finito. □

2.3 Specifiche di mutua esclusione generalizzate

Le specifiche di controllo per una rete di Petri possono essere:

- specifiche dinamiche che vincolano il linguaggio della rete;
- specifiche statiche che vincolano le marcature raggiungibili;
- specifiche qualitative che richiedono che la rete a ciclo chiuso goda di certe proprietà.

Una particolare forma di specifica statica è detta *specifica di mutua esclusione generalizzata o GMEC* (Generalized Mutual Exclusion Constraints) [2] e richiede di limitare la somma pesata delle marcature ottenute nella rete.

Le GMEC consentono di descrivere in maniera semplice le specifiche statiche: tali specifiche rivestono particolare importanza per il controllo di processi composti da più sottosistemi concorrenti che accedono in parallelo a un numero limitato di risorse.

Sarà inoltre introdotto il concetto di *posto monitor* come particolare struttura di supervisione che consente di imporre una GMEC su una rete di Petri.

2.3.1 Mutua esclusione e GMEC

Due condizioni logiche A_1 e A_2 sono dette in *mutua esclusione* se non possono essere contemporaneamente entrambe vere (esse potrebbero però essere entrambe false). Si assuma che una data condizione logica corrisponda in una rete P/T $\langle N, M_0 \rangle$ al fatto che un posto (supposto sano) sia marcato: ad esempio, la marcatura in cui $M(p_1) = 1$ rappresenta uno stato in cui la condizione A_1 è vera, mentre la marcatura in cui $M(p_2) = 1$ rappresenta uno stato in cui la condizione A_2 è vera. La specifica che due condizioni non debbano mai essere contemporaneamente vere si può rappresentare con il vincolo:

$$M(p_1) + M(p_2) \leq 1 \tag{2.1}$$

che dovrà essere verificato da tutte le marcature raggiungibili; infatti se i posti sono sani tale vincolo vieta solo le marcature per cui valga $M(p_1) = M(p_2) = 1$. Se la rete ha m posti, definendo

$$\boldsymbol{w} = [\ 1 \quad 1 \quad 0 \quad ... \quad 0\ ]^T$$

$$\boldsymbol{M} = [\ M(p_1) \quad M(p_2) \quad M(p_3) \quad ... \quad M(p_m)\ ]^T$$

vale anche

$$M(P_1) + M(p_2) = \boldsymbol{w}^T \boldsymbol{M}$$

e dunque posto $k = 1$ il vincolo (2.1) può scriversi:

$$\boldsymbol{w}^T \boldsymbol{M} \leq k. \tag{2.2}$$

Generalizzando, è possibile considerare vettori $\boldsymbol{w} = [\ w_1 \quad w_2 \quad w_3 \quad ... \quad w_m\ ]^T$

e scalari k che assumono valori interi e danno vincoli della forma

$$\boldsymbol{w}^T\boldsymbol{M} = w_1M(p_1) + w_2M(p_2) + w_3M(p_3) + ... + w_mM(p_m) \leq k \qquad (2.3)$$

dove $\boldsymbol{w} \in \mathbb{Z}^m$ e $k \in \mathbb{Z}$. In questo modo è possibile pesare in modo diverso la marcatura di posti diversi.

Definizione 2.3.1. *Sia $\langle N, M_0\rangle$ una rete marcata con m posti. Una specifica di mutua esclusione generalizzata o GMEC $(\boldsymbol{w}, k)$, costituita da un vettore $\boldsymbol{w} \in \mathbb{Z}^m$ e da uno scalare $k \in \mathbb{Z}$ definisce un insieme di marcature legali*

$$\mathcal{L}(\boldsymbol{w}, k) = \{\boldsymbol{M} \in \mathbb{N}^m \mid \boldsymbol{w}^T\boldsymbol{M} \leq k\}. \qquad (2.4)$$

L'insieme di *marcature raggiungibili e legali* è

$$M(N, \boldsymbol{M}_0, \boldsymbol{w}, k) = R(N, \boldsymbol{M}_0) \cap \mathcal{L}(\boldsymbol{w}, k). \qquad (2.5)$$

2.3.2 GMEC multiple

Quando si desidera imporre a un processo dato più GMEC $(\boldsymbol{w}_i, k_i)$ con $i = 1, ..., q$ è anche possibile raccogliere tutti questi vettori in una *GMEC multipla* $(\boldsymbol{W}, \boldsymbol{k})$ dove

$$\boldsymbol{W} = [\ \boldsymbol{w}_1 \quad \boldsymbol{w}_2 \quad ... \quad \boldsymbol{w}_q\ ]^T$$

è una matrice $m \times q$ le cui colonne sono i vettori che definiscono le singole GMEC e

$$\boldsymbol{k} = [\ k_1 \quad k_2 \quad ... \quad k_q\ ]^T$$

è un vettore $q \times 1$ che ha per elementi gli scalari k_i delle singole GMEC. Per tale GMEC multipla si definisce l'insieme di marcature legali:

$$\begin{aligned}\mathcal{L}(\boldsymbol{W}, \boldsymbol{k}) &= \{\boldsymbol{M} \in \mathbb{N}^m \mid \boldsymbol{W}^T\boldsymbol{M} \leq \boldsymbol{k}\} \\ &= \{\boldsymbol{M} \in \mathbb{N}^m \mid \boldsymbol{w}_1^T\boldsymbol{M} \leq k_1, \boldsymbol{w}_2^T\boldsymbol{M} \leq k_2, ...\boldsymbol{w}_q^T\boldsymbol{M} \leq k_q\} \\ &= \mathcal{L}(\boldsymbol{w}_1, k_1) \cap \mathcal{L}(\boldsymbol{w}_2, k_2) \cap \mathcal{L}(\boldsymbol{w}_q, k_q)\end{aligned}$$

e dunque una marcatura è legale per la GMEC multipla $(\boldsymbol{W}, \boldsymbol{k})$ se e solo se è legale per ogni singola GMEC $(\boldsymbol{w}_i, k_i)$.

Si ricordi che se una rete marcata $\langle N, \boldsymbol{M}_0\rangle$ è sana allora, data una qualunque specifica statica $\mathcal{L}$, è possibile determinare una GMEC multipla $(\boldsymbol{W}, \boldsymbol{k})$ tale che

$R(N, \boldsymbol{M}_0) \cap \mathcal{L} = R(N, \boldsymbol{M}_0) \cap \mathcal{L}(\boldsymbol{W}, \boldsymbol{k})$. Dunque almeno nel caso delle reti sane, ogni specifica statica ha una equivalente rappresentazione tramite GMEC.

2.3.3 Posti monitor

Le GMEC non consentono di descrivere tutte le specifiche di interesse, ma rivestono ugualmente un ruolo di grande importanza perchè il problema di controllo di reti di Petri con specifiche assegnate tramite GMEC è un problema che ammette una soluzione semplice e può essere affrontato con le tecniche dell'algebra lineare. La soluzione di tale problema è un posto monitor.

Data una GMEC $(\boldsymbol{w}, k)$ si vuole costruire un *supervisore*, cioè un *controllore* che garantisca che tale specifica non sia mai violata. Si supponga che la rete si trovi nella marcatura legale $\boldsymbol{M}$ e che lo scatto della transizione abilitata t porti il sistema a una marcatura $\boldsymbol{M}' = \boldsymbol{M} + \boldsymbol{C}(\cdot, t)$. Il supervisore dovrà permettere lo scatto di t se esso porta a una marcatura legale $\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{w}, k)$. Il supervisore dovrà invece impedire tale scatto, disabilitando la transizione t, se esso porta a una marcatura proibita $\boldsymbol{M}'$ non appartenente a $\mathcal{L}(\boldsymbol{w}, k)$.

Si supponga che ogni transizione sia controllabile, cioè che ogni transizione possa venir disabilitata dal supervisore se necessario. Il supervisore che permette di imporre una data GMEC prende la forma di un semplice posto detto *monitor*.

2.3.4 Monitor a ciclo chiuso

Definizione 2.3.2. *Sia dato un sistema $\langle N, \boldsymbol{M}_0 \rangle$ con matrice di incidenza $\boldsymbol{C}$ di dimensione $m \times n$ e sia $(\boldsymbol{w}, k)$ una GMEC che si desidera imporre. Si definisce* monitor *corrispondente a $(\boldsymbol{w}, k)$ il nuovo posto p_s che ha per matrice di incidenza il vettore riga $1 \times n$:*

$$\boldsymbol{C}_s = [C_s(t_1).....C_s(t_n)] = -\boldsymbol{w}^T \boldsymbol{C} \tag{2.6}$$

e marcatura iniziale

$$M_0(p_s) = k - \boldsymbol{w}^T \boldsymbol{M}_0. \tag{2.7}$$

Il posto monitor non ha cappi e dunque la matrice $\boldsymbol{C}_s$ è sufficiente per determinare gli archi pre e $post$ tra p_s e ogni transizione. Infatti, se $C_s(t) < 0$, il posto monitor p_s ha un arco pre verso t di molteplicità $Pre(p_s, t) = \mid C_s(t) \mid$ mentre,

se $C_s(t) > 0$, il posto monitor ha un arco post proveniente da t di molteplicità $Post(p_s, t) = C_s(t)$. Infine, se $C_s(t) = 0$, vale $Pre(p_s, t) = Post(p_s, t) = 0$.

L'aggiunta del posto monitor p_s alla rete $\langle N, \boldsymbol{M}_0 \rangle$ che descrive il processo determina una rete $\langle N_{cc}, \boldsymbol{M}_{cc,0} \rangle$ che descrive il sistema a ciclo chiuso (processo + supervisore) e ha una matrice d'incidenza

$$\boldsymbol{C}_{cc} = \left[\frac{\boldsymbol{C}}{\boldsymbol{C}_s} \right]$$

e marcatura iniziale

$$\boldsymbol{M}_{cc,0} = \left[\frac{\boldsymbol{M}}{\boldsymbol{M}_0(p_s)} \right].$$

Se si desidera imporre a un dato processo una GMEC multipla $(\boldsymbol{W}, \boldsymbol{k})$ dove

$$\boldsymbol{W} = [\ \boldsymbol{w}_1 \quad \boldsymbol{w}_2 \quad \ldots \quad \boldsymbol{w}_q\ ]^T$$

e

$$\boldsymbol{k} = [\ k_1 \quad k_2 \quad \ldots \quad k_q\ ]^T$$

si dovranno usare q monitor (tanti quanti sono i vincoli che costituiscono la GMEC multipla). L'aggiunta del posto p_s alla rete $\langle N, \boldsymbol{M}_0 \rangle$ che descrive il processo, determina una rete $\langle N_{cc}, \boldsymbol{M}_{cc,0} \rangle$ che descrive il sistema a ciclo chiuso (processo + supervisore) e ha matrice di incidenza

$$\boldsymbol{C}_{cc} = \left[\frac{\boldsymbol{C}}{\boldsymbol{C}_s} \right] = \left[\frac{\boldsymbol{C}}{-\boldsymbol{W}^T \boldsymbol{C}} \right]$$

e marcatura iniziale

$$\boldsymbol{M}_{cc,0} = \left[\frac{\boldsymbol{M}_0}{\boldsymbol{M}_{s,0}} \right] = \left[\frac{\boldsymbol{M}_0}{\boldsymbol{k} - \boldsymbol{W}^T \boldsymbol{M}_0} \right].$$

2.3.5 Monitor massimamente permissivi

In assenza di transizioni non controllabili il posto monitor che corrisponde a una data GMEC è un supervisore capace di imporre la specifica, cioè impedire di raggiungere marcature non legali, ed è anche *massimamente permissivo*, cioè blocca lo scatto di transizioni che portano a marcature non legali. In tal caso si parla di *monitor ottimo*.

Proposizione 2.3.1. *Dato il sistema $\langle N, \boldsymbol{M}_0 \rangle$ con matrice di incidenza $\boldsymbol{C}$, e una GMEC $(\boldsymbol{w}, k)$ che si desidera imporre, sia $\langle N_{cc}, \boldsymbol{M}_{cc,0} \rangle$ il sistema a ciclo chiuso con matrice di incidenza*

$$\boldsymbol{C}_{cc} = \left[\frac{\boldsymbol{C}}{-\boldsymbol{w}^T \boldsymbol{C}} \right]$$

e marcatura iniziale

$$\boldsymbol{M}_{cc,0} = \left[\frac{\boldsymbol{M}_0}{\boldsymbol{k} - \boldsymbol{w}^T \boldsymbol{M}_0} \right]$$

ottenuta aggiungendo il monitor corrispondente alla GMEC alla rete a ciclo aperto.
Data $\boldsymbol{M}_{cc} = [\boldsymbol{M}^T \mid M(p_s)]^T \in R(N_{cc}, \boldsymbol{M}_{cc})$ una generica marcatura raggiungibile del sistema a ciclo chiuso vale:

1. *$\boldsymbol{w}^T \boldsymbol{M} \leq k$, cioè la marcatura $\boldsymbol{M}$ del processo soddisfa la specifica;*
2. *data una qualunque transizione t se $\boldsymbol{M}[t\rangle \boldsymbol{M}'$ e $M(p_s) < Pre(p_s, t)$ vale $\boldsymbol{w}^T \boldsymbol{M}' > k$, cioè ogni qual volta una transizione t è abilitata dalla marcatura $\boldsymbol{M}$ del processo ma disabilitata dal monitor il suo scatto porta a una marcatura non legale.*

Dimostrazione. Si osservi che il vettore $\boldsymbol{x} = [\boldsymbol{w}^T 1]$ è un P-invariante (con coefficienti interi) del sistema a ciclo chiuso. Infatti vale

$$\boldsymbol{x}^T \boldsymbol{C}_{cc} = [\boldsymbol{w}^T 1] \left[\frac{\boldsymbol{C}}{-\boldsymbol{w}^T \boldsymbol{C}} \right] = \boldsymbol{w}^T \boldsymbol{C} - \boldsymbol{w}^T \boldsymbol{C} = 0^T.$$

In base alle proprietà dei P-invarianti per ogni marcatura raggiungibile del sistema a ciclo chiuso $\boldsymbol{M}_{cc} = [\boldsymbol{M}^T \mid M(p_s)]^T$ deve valere $\boldsymbol{x}^T \boldsymbol{M}_{cc} = x^T \boldsymbol{M}_{cc,0}$ ossia

$$\boldsymbol{w}^T \boldsymbol{M} + M(p_s) = \boldsymbol{x}^T \boldsymbol{M}_{cc} = \boldsymbol{x}^T \boldsymbol{M}_{cc,0} = \boldsymbol{x}^T \boldsymbol{M}_0 + k - \boldsymbol{x}^T \boldsymbol{M}_0 = k. \quad (2.8)$$

Per dimostrare il primo punto si osservi che in base alla relazione (2.8), per ogni marcatura raggiungibile del sistema a ciclo chiuso $\boldsymbol{M}_{cc} \in R(N_{cc}, \boldsymbol{M}_{cc,0})$ vale $\boldsymbol{w}^T \boldsymbol{M} = k - M(p_s) \leq k$ e dunque $\boldsymbol{M}$ è legale.
Per dimostrare il secondo punto, sia t una transizione tale che $\boldsymbol{M}[t\rangle \boldsymbol{M}'$ e dunque tale che $\boldsymbol{M}' = \boldsymbol{M} + \boldsymbol{C}(\cdot, t)$. Dalla definizione di posto monitor è facile capire che vi sarà un arco *pre* da p_s a t se e solo se $C_s(t) < 0$ e in tal caso la molteplicità dell'arco vale $Pre(p_s, t) = -C_s(t) = \boldsymbol{w}^T \boldsymbol{C}_s(\cdot, t)$. Dunque se la transizione è disabilitata dal monitor p_s vale $M(p_s) < Pre(p_s, t) = \boldsymbol{w}^T \boldsymbol{C}(\cdot, t)$ e dunque

$\boldsymbol{w}^T\boldsymbol{M}' = \boldsymbol{w}^T[\boldsymbol{M} + \boldsymbol{C}(.,t)] = \boldsymbol{w}^T\boldsymbol{M} + \boldsymbol{w}^T\boldsymbol{C}(.,t) > \boldsymbol{w}^T\boldsymbol{M} + M(p_s)$ dove l'ultima uguaglianza deriva dalla (2.8) e dunque $\boldsymbol{M}'$ non è legale. □

Nella dimostrazione precedente si è osservato che per ogni marcatura raggiungibile del sistema a ciclo chiuso $\boldsymbol{M}_{cc} = [\boldsymbol{M}^T \mid M(p_s)]^T$ vale $M(p_s) = k - \boldsymbol{w}^T\boldsymbol{M}$: sostanzialmente la marcatura del posto monitor indica di quanto la marcatura pesata dei posti della rete dista dal valore massimo ammissibile k. In questo senso, si può pensare alla marcatura del posto monitor come a una variabile *slack* che assume il valore zero quando il vincolo $\boldsymbol{w}^T\boldsymbol{M} \leq k$ imposto dalla GMEC vale con l'uguaglianza, mentre assume valori positivi quando il vincolo vale con la disuguaglianza stretta.

2.3.6 Realizzabilità di un posto monitor

Proposizione 2.3.2. *Dato il sistema $\langle N, \boldsymbol{M}_0\rangle$ sia $(\boldsymbol{w}, k)$ una GMEC che si desidera imporre. Il valore della marcatura iniziale del posto monitor corrispondente alla GMEC $M_0(p_s) = k - \boldsymbol{w}^T\boldsymbol{M}_0$ è non negativo se e solo se $\boldsymbol{M}_0 \in \mathcal{L}(\boldsymbol{w}, k)$, cioè se e solo se la marcatura iniziale $\boldsymbol{M}_0$ del processo è legale.*

Dimostrazione. Si osservi che vale

$$M_0(p_s) \geq 0 \iff k - \boldsymbol{w}^T\boldsymbol{M}_0 \geq 0 \iff \boldsymbol{w}^T\boldsymbol{M}_0 \leq k \iff \boldsymbol{M}_0 \in \mathcal{L}(\boldsymbol{w}, k)$$

che dimostra il risultato enunciato. □

Se la marcatura iniziale $\boldsymbol{M}_0$ del processo è legale, il posto monitor è fisicamente realizzabile e la GMEC può essere imposta. Se viceversa, $\boldsymbol{M}_0$ non fosse legale, non vi è modo di imporre la GMEC poichè essa è già inizialmente violata.

Si conlcude tale paragrafo osservando alcuni vantaggi derivanti dall'uso dei posti monitor come supervisori.

La struttura di controllo è molto semplice, poichè consiste in un numero di posti pari alle GMEC che costituiscono la specifica, inoltre non richiede di modificare la struttura della rete a ciclo aperto o di aggiungere alcuna transizione.

Il progetto del monitor non richiede l'analisi dello spazio di stato della rete, che può essere anche infinito, ma si basa su semplici prodotti matriciali.

Nota 2.3.1. *Viene introdotta ora una definizione relativa alle GMEC che risulterà utile in seguito, quando verrà esposto il problema preso in esame da questa tesi. Si definisce* supporto *di* $(\boldsymbol{w}, k)$ *l'insieme* $Q_{\boldsymbol{w}} = \{p \in P | \boldsymbol{w}(p) \neq 0\}$. *Un insieme di GMEC* $(\boldsymbol{W}, \boldsymbol{k})$ *con* $\boldsymbol{W} = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, ...\boldsymbol{w}_{n_c}^T]^T$ *e* $\boldsymbol{k} = [k_1, k_2, ...k_{n_c}]^T$*, definisce l'insieme di marcature legali* $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{W} \cdot \boldsymbol{m} \leq \boldsymbol{k}\}$. *Per cui si chiama* supporto *di* $(\boldsymbol{W}, \boldsymbol{k})$ *l'insieme*

$$Q_{\boldsymbol{W}} = \left(\bigcup_{j=1}^{n_c} Q_{\boldsymbol{w}_j} \right).$$

Capitolo 3

Approccio decentralizzato

In questo capitolo verrà presentata nei dettagli la formulazione del problema della determinazione di un insieme di monitor decentralizzati data una GMEC e le due possibili soluzioni presentate in [1]. Prima di enunciare il problema verranno introdotte alcune definizioni geometriche utili a comprendere meglio il suo significato fisico. Successivamente si formulerà il problema e si distingueranno due differenti approcci:

caso A: massimizzazione della cardinalità delle marcature basandosi sulla programmazione matematica e su determinate assunzioni;

caso B: risoluzione del problema usando degli algoritmi iterativi che verranno implementati tramite Matlab. In questo caso non sono richieste le assunzioni fatte nel *caso A*.

3.1 Definizioni geometriche

Prima di procedere alla formulazione del problema è necessario introdurre i concetti di poliedro convesso, box e ipercubo.

Un poliedro convesso è un insieme di punti in cui, presi due punti x_1 e x_2, il segmento che li congiunge è interamente contenuto nell'insieme.

Definizione 3.1.1. *Sia $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^d \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ un poliedro convesso dove $\boldsymbol{A}$ è una matrice reale $r \times d$ e $\boldsymbol{b}$ è un vettore reale di dimensione d. Un punto $\hat{\boldsymbol{x}}$ tale che $\boldsymbol{A}\hat{\boldsymbol{x}} < \boldsymbol{b}$ è un punto interno del poliedro $\mathcal{P}$.*

Un poliedro che ha un punto interno è detto dimensionalmente pieno; in caso contrario è incapsulato in uno spazio affine di dimensione inferiore.

Definizione 3.1.2. *Un box è un insieme definito da due vettori reali $\boldsymbol{l}$ e $\boldsymbol{u}$ di dimensione d, tali che*

$$\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) = \{\boldsymbol{x} \in \mathbb{R}^d \mid \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}.$$

Se $\boldsymbol{x} \in \mathbb{N}^d$, $\boldsymbol{l} = \boldsymbol{0}$ e $\boldsymbol{u} \in \mathbb{N}^d$, $\mathcal{B}(\boldsymbol{0}, \boldsymbol{u})$ si chiama box intero e si indica semplicemente come $\mathcal{I}(\boldsymbol{u})$.

Definizione 3.1.3. *Un ipercubo è un box tale che $\boldsymbol{u} = \boldsymbol{l} + \lambda\boldsymbol{e}$, dove λ è uno scalare e $\boldsymbol{e}$ è un vettore di dimensione d i cui elementi sono pari a 1; un ipercubo intero è un box $\mathcal{I}(\boldsymbol{u})$ tale che $\boldsymbol{u} = \lambda\boldsymbol{e}$ dove λ è uno scalare intero positivo.*

La cardinalità di un insieme $\mathcal{S}$ viene indicata con $|\mathcal{S}|$.

Definizione 3.1.4. *Un box intero $\mathcal{I}(\boldsymbol{u}) \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ è un box intero interno a massima cardinalità se non esiste un box intero interno $\mathcal{I}(\tilde{\boldsymbol{u}}) \neq \mathcal{I}(\boldsymbol{u})$ tale che $\mathcal{I}(\tilde{\boldsymbol{u}}) \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ e $|\mathcal{I}(\boldsymbol{u})| < |\mathcal{I}(\tilde{\boldsymbol{u}})|$.*

La precedente definizione conduce a due importanti risultati:

- la cardinalità di un box intero interno $\mathcal{I}(\boldsymbol{u})$ è uguale a $\prod_{p \in P} (\boldsymbol{u}(p) + 1)$ poichè ogni lato include tutti i numeri interi compresi tra 0 e $\boldsymbol{u}(p)$;
- la cardinalità di un ipercubo intero $\mathcal{I}(\tau\boldsymbol{e})$ avente un vertice nell'origine e lato uguale a τ è pari a $(\tau + 1)^m$.

Definizione 3.1.5. *Sia $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^d \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ un poliedro convesso. Un box $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) \subseteq \mathcal{P}$ è un box interno massimale in $\mathcal{P}$ se non esiste un altro box interno $\mathcal{B}(\tilde{\boldsymbol{l}}, \tilde{\boldsymbol{u}}) \neq \mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ tale che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) \subsetneq \mathcal{B}(\tilde{\boldsymbol{l}}, \tilde{\boldsymbol{u}}) \subseteq \mathcal{P}$.*

Perciò si può concludere dicendo che un box $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ è un box massimale interno a un poliedro convesso $\mathcal{P}$ se non esiste nessuna direzione in cui i lati di tale box possano essere aumentati senza fuoriuscire dal poliedro $\mathcal{P}$.

3.2 Esposizione del problema

Data una rete P/T marcata $\langle N, \boldsymbol{M}_0 \rangle$ in cui $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ e $\boldsymbol{M}_0$ marcatura iniziale, sia $(\boldsymbol{W}, \boldsymbol{k})$ una specifica globale che si desidera imporre. L'insieme di marcature raggiungibili è definito come $R(N, M_0)$ e l'insieme di marcature legali è definito come $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$. L'obiettivo principale della teoria dei supervisori di controllo è quello di permettere al sistema di raggiungere tutte le marcature in $R(N, M_0) \cap \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$; tutte le altre marcature sono illegali e devono essere proibite.

In [1] si assume che l'insieme dei posti P sia partizionato in ν sottoinsiemi $P_1, \ldots,$ P_ν e devono essere realizzati ν controllori decentralizzati, ognuno dei quali può prendere decisioni solo sulla base delle marcature del sottoinsieme di posti.

Definito

$$\mathcal{M}_d \equiv \cap_{i=1}^{\nu} \mathcal{M}^i \tag{3.1}$$

come l'insieme di marcature legali decentralizzate, dove

$$\mathcal{M}^i \equiv \mathcal{M}(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)}), \qquad i = 1, \ldots, \nu, \tag{3.2}$$

il controllo decentralizzato consiste nel determinare l'insieme di GMEC decentralizzate $(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)})$ per $i = 1, \ldots, \nu$, che consentono di massimizzare la cardinalità dell'insieme $R(N, M_0) \cap \mathcal{M}_d$, garantendo che l'insieme di marcature raggiungibili e marcature legali decentralizzate siano ancora comprese tra le marcature legali $R(N, M_0) \cap \mathcal{M}_d \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$; quindi le marcature che sono raggiungibili sotto il controllo decentralizzato sono legali.

Data una generica rete P/T $\langle N, M_0 \rangle$, una generica GMEC globale $(\boldsymbol{W}, \boldsymbol{k})$ e una partizione di posti, trovare una procedura che determini una soluzione ottimale per il problema di controllo decentralizzato, è un compito piuttosto complesso poichè conduce a un'enumerazione dello spazio di raggiungibilità. Perciò una soluzione efficiente propone come obiettivo quello di massimizzare la cardinalità dell'insieme $\mathcal{M}_d$, piuttosto che massimizzare la cardinalità dell'insieme $R(N, M_0) \cap \mathcal{M}_d$. In questo modo possono essere trovate soluzioni corrette al problema originale anche se non necessariamente ottimali.

La seguente proposizione dà un criterio per calcolare ottimamente, in termini di massimalità dell'insieme $\mathcal{M}_d$, i coefficienti delle matrici $\boldsymbol{W}^{(i)}$, lasciando quindi come uniche incognite del problema i vettori $\boldsymbol{k}^{(i)}$.

Proposizione 3.2.1. *Siano $(\boldsymbol{w}, k)$ una GMEC singola globale e $(\bar{\boldsymbol{w}}^{(i)}, \bar{k}^{(i)})$, $i = 1, \ldots, \nu$ un insieme di ν GMEC decentralizzate. Sia $(\boldsymbol{w}^{(i)}, k^{(i)})$ per $i = 1, \ldots, \nu$*

un insieme di ν GMEC decentralizzate dove $\boldsymbol{w}^{(i)}$ è definito come segue

$$\boldsymbol{w}^{(i)}(p) = \begin{cases} \boldsymbol{w}(p) & \textit{se } p \in P_i \\ 0 & \textit{altrimenti}, \end{cases} \tag{3.3}$$

per $p \in P$ e per $i = 1, \ldots, \nu$. Esiste sempre un insieme di $k^{(i)}$ tale che

$$\mathcal{M}_d \supseteq \bar{\mathcal{M}}_d \tag{3.4}$$

dove

$$\mathcal{M}_d = \cap_{i=1}^{\nu} \mathcal{M}^{(i)} = \cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{w}^{(i)}, k^{(i)}), \bar{\mathcal{M}}_d = \cap_{i=1}^{\nu} \bar{\mathcal{M}}^{(i)} = \cap_{i=1}^{\nu} \mathcal{M}(\bar{\boldsymbol{w}}^{(i)}, \bar{k}^{(i)}) \tag{3.5}$$

e $\mathcal{M}_d \subseteq \mathcal{M}(\boldsymbol{w}, k)$.

La dimostrazione di questa proposizione è presentata in [1]. Tale proposizione conduce a un importante risultato poichè afferma che la scelta ottimale dei pesi dei vettori delle GMEC decentralizzate, in termini di massimalità dell'insieme $\mathcal{M}_d$, consiste nel prendere tali valori uguali alla proiezione dei pesi dei vettori della GMEC globale sugli insiemi P_i. Questa condizione risulta difficilmente utilizzabile nel caso in cui non vengano fatte assunzioni sui pesi delle matrici.

Si distinguono due casi che conducono a due formulazioni differenti del problema:

1. *caso A*, assume che i pesi siano positivi, $\boldsymbol{W} \geq \mathbf{0}$ e $\boldsymbol{k} \geq \mathbf{0}$;
2. *caso B*, assume vincoli rilassati rispetto alla formulazione originale.

3.3 Caso A: approccio con programmazione matematica

L'obiettivo di tale approccio è quello di massimizzare la cardinalità dell'insieme di marcature legali decentralizzate $\mathcal{M}_d$ con le seguenti assunzioni:

- pesi delle GMEC positivi ($\boldsymbol{W} \geq \mathbf{0}$ e $\boldsymbol{k} \geq \mathbf{0}$);
- suddivisione dei posti di tipo singleton.

Senza le suddette assunzioni, tale problema, risolvibile con la programmazione matematica, non sarebbe in grado di fornire una soluzione.

In [1] si propone di considerare ogni GMEC singola $(\boldsymbol{w}, k)$, che costituisce la GMEC globale $(\boldsymbol{W}, \boldsymbol{k})$, separatamente, al fine di analizzare ogni vincolo distintamente l'uno dall'altro.

I vettori $\boldsymbol{w}^{(i)}$ possono essere definiti usando la relazione (3.3) come la proiezione del vettore w sui supporti del posto P_i della GMEC decentralizzata; l'unica incognita del problema risulta essere la costante $k^{(i)}$.
La soluzione proposta in [1] si basa su considerazioni geometriche che, nel caso in cui il vincolo $\boldsymbol{m} \in \mathbb{N}^m$ sia rilassato a $\boldsymbol{m} \in (\mathbb{R}_0^+)^m$, conducono a un criterio ottimale. La soluzione si basa sul seguente risultato.

Lemma 3.3.1. *Si assuma che*

$$\sum_{i=1}^{m} w_i x_i \leq b, \quad w_i, x_i, b \in \mathbb{R}_0^+$$

sia una regione nello spazio m-dimensionale. Il volume generalizzato di questa regione indicato con V_m, è uguale a

$$V_m(b, w_1, \ldots, w_m) = \frac{1}{m!} \frac{b^m}{\prod_{i=1}^{m} w_i}.$$

Non si ripete in questo contesto la dimostrazione di tale risultato poichè esso è presente in [1].

Tale lemma conduce a formulare la seguente proposizione.

Proposizione 3.3.1. *Sia $(\boldsymbol{w}, k)$ una GMEC globale su una data rete e sia $P_1, \ldots, P_\nu$ una data partizione di posti. Si assuma che i vettori $\boldsymbol{w}^{(i)}$ della i-esima GMEC decentralizzata siano definiti come nell'equazione (3.3) e che il vincolo di integrità sulla marcatura $\boldsymbol{m} \in \mathbb{N}^m$, sia rilassato a $\boldsymbol{m} \in (\mathbb{R}_0^+)^m$. Sia $k^{(i)}$, $i = 1, \ldots, \nu$ una soluzione del seguente problema di programmazione non lineare intera*

$$\begin{cases} \max & \prod_{i=1}^{\nu} (k^{(i)})^{n_i} \\ \\ \textit{s.t.} & \sum_{i=1}^{\nu} k^{(i)} \leq k \qquad (a) \\ & k^{(i)} \in \mathbb{N}, \;\; i = 1, \ldots, \nu. \end{cases} \tag{3.6}$$

Il volume generalizzato dell'insieme convesso $\cap_{i=1}^{\nu} \mathcal{M}_R(\boldsymbol{w}^{(i)}, k^{(i)})$, dove

$$\mathcal{M}_R(\boldsymbol{w}^{(i)}, k^{(i)}) = \{\boldsymbol{m} \in (\mathbb{R}_0^+)^m \;\mid\; \boldsymbol{w}^{(i)} \boldsymbol{m} \leq k^{(i)}\}$$

è l'insieme di marcature rilassate in accordo con la GMEC $(\boldsymbol{w}^{(i)}, k^{(i)})$*, è maggiore o uguale al volume di un altro insieme convesso* $\cap_{i=1}^{\nu} \mathcal{M}_R(\boldsymbol{w}^{(i)}, \bar{k}^{(i)}) \subseteq \mathcal{M}_R(\boldsymbol{w}, k)$ *con* $\bar{k}^{(i)} \neq k^{(i)}$ *per alcuni* $i = 1, \ldots, \nu$ *e* $\mathcal{M}_R(\boldsymbol{w}, k) = \{\boldsymbol{m} \in (\mathbb{R}_0^+)^m \mid \boldsymbol{w}\boldsymbol{m} \leq k\}$.

Dimostrazione. Dal Lemma (3.3.1) ogni GMEC decentralizzata $(\boldsymbol{w}^{(i)}, k^{(i)})$ definisce una regione convessa $\mathcal{M}_R(\boldsymbol{w}^{(i)}, k^{(i)})$ nello spazio n_i dimensionale avente un volume uguale a

$$\frac{1}{n_i!} \frac{(k^{(i)})^{n_i}}{\prod_{p \in P_i} w^{(i)}(p)}.$$

Inoltre la regione definita dall'intersezione di $\mathcal{M}_R(\boldsymbol{w}^{(i)}, k^{(i)})$ per $i = 1, \ldots, \nu$ ha un volume uguale al prodotto di ν volumi.

Per massimizzare l'intero volume si vuole che $k^{(i)}$ sia tale che $\prod_{i=1}^{\nu} (k^{(i)})^{n_i}$ sia massimo, essendo gli altri termini costanti rispetto a $k^{(i)}$. □

Esempio 3.3.1. *Sia* $(\boldsymbol{w}, k)$ *una GMEC globale di una rete P/T dove* $\boldsymbol{w} = [3 \;\; 2]^T$ *e* $k = 5$*. Si assuma* $P = \{p_1, p_2\}$ *e* $P_1 = \{p_1\}$*,* $P_2 = \{p_2\}$*. Le GMEC decentralizzate sono* $3m_1 \leq k^{(1)}$ *e* $2m_2 \leq k^{(2)}$ *dove le costanti* $k^{(1)}$ *e* $k^{(2)}$ *devono essere determinate.*

Si assuma che l'integrità sul vincolo $\boldsymbol{m}$ *sia rilassata per cui* $\boldsymbol{m} \in (\mathbb{R}_0^+)^m$*. In tal caso le GMEC decentralizzate* $(w^{(1)}, k^{(1)})$ *e* $(w^{(2)}, k^{(2)})$ *che massimizzano il volume* $\mathcal{M}_R(w^{(1)}, k^{(1)}) \cap \mathcal{M}_R(w^{(2)}, k^{(2)})$*, possono essere trovate risolvendo il problema di programmazione non lineare*

$$\begin{cases} \max & k^{(1)} k^{(2)} \\ s.t. & k^{(1)} + k^{(2)} = 5 \\ & k^{(1)}, k^{(2)} \in \mathbb{N}. \end{cases}$$

Tale problema ha due differenti soluzioni:

1. $\bar{k}^{(1)} = 3$*,* $\bar{k}^{(2)} = 2$*,*
2. $\tilde{k}^{(1)} = 2$*,* $\tilde{k}^{(2)} = 3$*.*

In Figura (3.1) si può vedere che l'insieme $\mathcal{M}_R(\boldsymbol{w}, k)$ *è un triangolo di base* $5/3$ *e altezza* $5/2$*, l'insieme* $\mathcal{M}_R(\boldsymbol{w}^{(1)}, \bar{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \bar{k}^{(2)})$ *è un quadrato di area unitaria e l'insieme* $\mathcal{M}_R(\boldsymbol{w}^{(1)}, \tilde{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \tilde{k}^{(2)})$ *è un rettangolo di base* $2/3$ *e altezza* $3/2$*.*

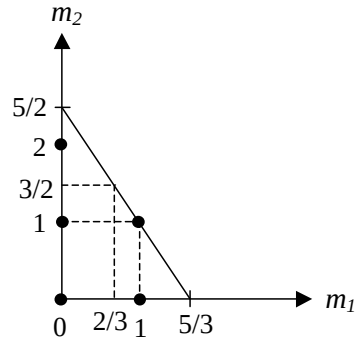

Figura 3.1: Esempio numerico 3.3.1

I volumi degli insiemi convessi $\mathcal{M}_R(\boldsymbol{w}^{(1)}, \bar{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \bar{k}^{(2)})$ *e* $\mathcal{M}_R(\boldsymbol{w}^{(1)}, \tilde{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \tilde{k}^{(2)})$ *sono gli stessi, essendo entrambe le GMEC decentralizzate ottimali rispetto all'obiettivo considerato.* ■

La Proposizione (3.3.1) non può essere estesa al caso in cui $\boldsymbol{m} \in \mathbb{N}^m$. Per provare tale affermazione si consideri il seguente esempio.

Esempio 3.3.2. Si consideri di nuovo l'Esempio (3.3.1) con il vincolo $\boldsymbol{m} \in \mathbb{N}^2$.

Come si nota dalla Figura (3.1) vi sono solo 5 marcature intere che risultano in accordo con le GMEC globali $(\boldsymbol{w}, k)$: infatti l'insieme di marcature legali è

$$\mathcal{M}(\boldsymbol{w}, k) = \{[0\ 0], [0\ 1], [1\ 0], [1\ 1], [0\ 2]\}.$$

Inoltre $\mathcal{M}(\boldsymbol{w}^{(1)}, \bar{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \bar{k}^{(2)}) = \{[0\ 0], [0\ 1], [1\ 0], [1\ 1]\}$.
Invece $\mathcal{M}(\boldsymbol{w}^{(1)}, \tilde{k}^{(1)}) \cap \mathcal{M}_R(\boldsymbol{w}^{(2)}, \tilde{k}^{(2)}) = \{[0\ 0], [1\ 0]\}$, mette in evidenza che la seconda soluzione non è ottimale. ■

Questo esempio dimostra quanto annunciato precedentemente: un problema di programmazione matematica giunge a una soluzione, anche non ottimale, se vengono fatte assunzioni generali sulla suddivisione dell'insieme di posti. Tuttavia, se tutti i sottoinsiemi P_i sono singleton e contengono un solo posto, si può dimostrare il risultato seguente che fornisce un criterio ottimale per selezionare le GMEC decentralizzate.

Proposizione 3.3.2. *Sia* $(\boldsymbol{w}, k)$ *una GMEC su una rete. Dato* $P_i = \{p_i\}$ *per* $i = 1, \ldots, \nu$ *(allora* $\nu = m$*) e si assuma che* $\boldsymbol{w}^{(i)}$ *siano scelti come in (3.3). Sia* $k^{(i)}$*,* $i = 1, \ldots, \nu$ *una soluzione del seguente problema di programmazione intera*

non lineare

$$\begin{cases} \max \quad \prod_{i=1}^{\nu} k^{(i)} & \\ \\ \textit{s.t.} \quad \sum_{i=1}^{\nu} k^{(i)} = k & (a) \\ \quad \dfrac{k^{(i)}}{w(p_i)} \in \mathbb{N} \;\; i = 1, \ldots, \nu & (b) \\ \quad k^{(i)} \in \mathbb{N}, \;\; i = 1, \ldots, \nu. & \end{cases} \tag{3.7}$$

La cardinalità dell'insieme $\cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{w}^{(i)}, k^{(i)})$ *è piu grande o uguale alla cardinalità di un altro insieme* $\cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{w}_i, \bar{k}^{(i)}) \subseteq \mathcal{M}(\boldsymbol{w}, k)$ *con* $\bar{k}^{(i)} \neq k^{(i)}$.

Dimostrazione. La validità del seguente enunciato deriva dalla Proposizione (3.3.1) e grazie al vincolo (b), si rifiutano tutte quelle GMEC decentralizzate che portano a poliedri i cui vertici non sono caratterizzati da coordinate intere, cioè poliedri il cui volume è massimo ma il cui numero di punti interni interi non è massimo. □

Esempio 3.3.3. *Si consideri un altro caso dell'Esempio (3.3.1). Il problema di programmazione intera (3.7) fornisce la soluzione* $k^{(1)} = 3$ *e* $k^{(2)} = 2$*, allora* $\mathcal{M}(\boldsymbol{w}_1, k^{(1)}) \cap \mathcal{M}(\boldsymbol{w}_2, k^{(2)}) = \{[0\ 0], [0\ 1], [1\ 0], [1\ 1]\}$ *fornisce la soluzione ottimale.*

3.4 Caso B: posti singleton

Il secondo approccio considerato ha l'obiettivo di determinare un insieme di marcature legali decentralizzato che sia massimale e in grado di garantire una buona fairness tra i posti.

La soluzione proposta in [1] per questo problema non richiede nessun'assunzione sul peso dei coefficienti della matrice $\boldsymbol{W}$ e del vettore $\boldsymbol{k}$, ma viene raggiunta in due step: in un primo momento si assume che tutti i posti siano singleton e si risolve un problema di programmazione lineare intero al fine di ricercare il centro del box $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ incluso in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$; in un secondo momento a partire dai risultati raggiunti si trova un box intero $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{l} \leq \boldsymbol{m} \leq \boldsymbol{u}\}$ che sia massimale e tale che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$, cioè è incluso nell'insieme di marcature legali definite dalla GMEC globale. La ricerca di tale box avviene per mezzo di due algoritmi la cui implementazione è l'oggetto di studio e risoluzione di questa tesi.

Si può riassumere l'approccio eseguito nei seguenti passi.

1. Si determina un punto interno $\boldsymbol{c} \in \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

2. Si definisce un nuovo sistema di assi coordinati con centro in $\boldsymbol{c}$. Considerando il punto come centro del nuovo sistema d'assi, le coordinate subiranno una traslazione del tipo $\boldsymbol{m}' = \boldsymbol{m} - \boldsymbol{c}$ e l'insieme di marcature legali $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ è trasformato nell'insieme $\mathcal{M}(\tilde{\boldsymbol{W}}', \tilde{\boldsymbol{k}}') = \{\boldsymbol{m}' \in \mathbb{N}^m \mid \tilde{\boldsymbol{W}}' \cdot \boldsymbol{m}' \leq \tilde{\boldsymbol{k}}'\}$ con $\tilde{\boldsymbol{W}}' = \boldsymbol{W}$ e $\tilde{\boldsymbol{k}}' = \boldsymbol{k} - \boldsymbol{W} \cdot \boldsymbol{c}$. Il vettore $\boldsymbol{m}$ rappresenta la marcatura della rete e può assumere solo valori positivi; ciò conduce a imporre un ulteriore vincolo, $-\boldsymbol{m}' \leq \boldsymbol{c}$. Si denoti con $(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$ la GMEC risultante dalla GMEC $(\boldsymbol{W}, \boldsymbol{k})$ e dai corrispondenti vincoli di non negatività.

3. Per mezzo di due differenti algoritmi che verranno esposti in seguito e la cui implementazione è l'obiettivo di questo lavoro di tesi, si trova un box intero interno massimale in $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$.

4. Come ultimo passo si esegue una traslazione al sistema di coordinate originale $\boldsymbol{m} = \boldsymbol{m}' + \boldsymbol{c}$ che permette di determinare le GMEC decentralizzate per il sistema originale.

3.4.1 Definizioni introduttive

Si denoti con $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ un generico insieme convesso contenente l'origine tale che $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b}) = \{\boldsymbol{m} \in \mathbb{R}^m \mid \boldsymbol{A}\boldsymbol{m} \leq \boldsymbol{b}\}$ dove $\boldsymbol{A} \in \mathbb{Z}^{n_c \times m}$ e $\boldsymbol{b} \in \mathbb{N}^{n_c}$. Si indichi con $\boldsymbol{a}_i$ la riga i-esima della matrice $\boldsymbol{A}$ e con b_i la i-esima componente del vettore $\boldsymbol{b}$. Si consideri $\boldsymbol{a}_i$ un vettore riga di dimensione m, il quale indica il numero di posti, perciò $a_i(p)$ indica la componente del vettore $\boldsymbol{a}_i$ relativa al posto p.

Definizione 3.4.1. *Un box intero $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) \subseteq \mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ è un box intero interno massimale se non esiste un altro box interno $\mathcal{B}(\tilde{\boldsymbol{l}}, \tilde{\boldsymbol{u}}) \neq \mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ tale che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u}) \subsetneq \mathcal{B}(\tilde{\boldsymbol{l}}, \tilde{\boldsymbol{u}}) \subseteq \mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$.* ■

Si noti che il box intero interno massimale è generalmente non unico, ciò indica che si possono avere diversi box con le caratteristiche richieste.

Definizione 3.4.2. *Si consideri un box intero $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ incluso in un insieme convesso $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$, dove entrambi $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ e $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ contengono l'origine. E' possibile definire un vettore estremo di $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ il vettore $\boldsymbol{x}$ di dimensione n_c come segue*

$$x_i(p) = \begin{cases} l(p), & \textit{se } a_i(p) \leq 0, \\ u(p), & \textit{se } a_i(p) > 0, \end{cases} \qquad i = 1, \ldots, n_c. \tag{3.8}$$

■

Proposizione 3.4.1. *Dato un box intero $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ incluso in un insieme convesso $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ dove sia $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ che $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ contengono l'origine, il box $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ è un box intero interno massimale in $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ se e solo se $\forall p \in P$*

$$0 \leq \min_{i \in \{1,\ldots,n_c\}} \frac{b_i - \sum_{p \in P} a_i(p) x_i(p)}{|a_i(p)|} < 1 \tag{3.9}$$

dove $\boldsymbol{x}$ è definito come nell'equazione (3.8).

Dimostrazione. La prima parte della disuguaglianza è banale perchè assicura il soddisfacimento dei vincoli dal momento che $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ contiene l'origine.

Si discute ora la seconda parte della disuguaglianza.
(Se) Sia p un posto in P e sia $\bar{\imath}$ il corrispondente valore di indice i che massimizza la relazione (3.9).
Ne deriva

$$\frac{b_{\bar{\imath}} - \sum_{p \in P} a_{\bar{\imath}}(p) x_{\bar{\imath}}(p)}{|a_{\bar{\imath}}(p)|} < 1 \quad \Rightarrow b_{\bar{\imath}} - \sum_{p \in P} a_{\bar{\imath}}(p) x_{\bar{\imath}}(p) < |a_{\bar{\imath}}(p)|.$$

Ora, dall'assunzione che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ contenga l'origine, allora $\boldsymbol{l} \leq \mathbf{0}$ e $\boldsymbol{u} \geq \mathbf{0}$. Inoltre

- se $a_{\bar{\imath}}(p) < 0$ allora $\boldsymbol{a}_{\bar{\imath}} \cdot \tilde{\boldsymbol{x}}_{\bar{\imath}} > b_{\bar{\imath}}$ dove $\tilde{x}_{\bar{\imath}}(p_j) = l_{\bar{\imath}}(p_j)$ per tutti $p_j \neq p$, e $\tilde{x}_{\bar{\imath}}(p) = l(p) - 1$;
- se $a_{\bar{\imath}}(p) \geq 0$ allora $\boldsymbol{a}_{\bar{\imath}} \cdot \tilde{\boldsymbol{x}}_{\bar{\imath}} > b_{\bar{\imath}}$ dove $\tilde{x}_{\bar{\imath}}(p_j) = u_{\bar{\imath}}(p_j)$ per tutti $p_j \neq p$, e $\tilde{x}_{\bar{\imath}}(p) = u(p) + 1$.

Questo significa che se il lower bound e l'upper bound su p è aumentato o diminuito di un'unità, ciò porterebbe alla violazione della $\bar{\imath}$-iesima GMEC. Poichè questo è vero per tutti i posti $p \in P$, si conclude che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ è un box intero massimale in $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$.
(Solo se) Ciò si può dimostrare per assurdo. Si assuma che $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ sia un box intero interno massimale in $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$, ma $\exists\, p \in P$ così che

$$\min_{i \in \{1,\ldots,n_c\}} \frac{b_i - \sum_{p \in P} a_i(p) x_i(p)}{|a_i(p)|} \geq 1.$$

Ciò implica $\forall i \in \{1, \ldots, n_c\}$, $b_i - \boldsymbol{a}_i \cdot \boldsymbol{x}_i \geq |a_i(p)|$. Dunque, dato un posto arbitrario $p \in P$ si definisce un vettore $\tilde{\boldsymbol{x}}_i$ come nell'enunciato precedente che soddisfi tutti i vincoli, così che $\boldsymbol{a}_i \cdot \tilde{\boldsymbol{x}}_i \leq b_i$ per $i = 1, \ldots, n_c$. Ciò è una contraddizione. □

In altre parole la proposizione precedente significa che un box intero $\mathcal{B}(\boldsymbol{l}, \boldsymbol{u})$ è massimale se e solo se in ogni direzione esiste almeno un vincolo che viene saturato.

Infine, la proposizione seguente fornisce un criterio per determinare l'ipercubo massimale intero in $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ con centro nell'origine.

Proposizione 3.4.2. *Sia $\bar{\mathcal{C}} = \mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ un insieme convesso contenente l'origine e dunque $\boldsymbol{b} \geq \boldsymbol{0}$. Sia*

$$\tau(\bar{\mathcal{C}}) = \max \{\tau \in \mathbb{N} \mid \mathcal{B}(-\tau \boldsymbol{e}, \tau \boldsymbol{e}) \subseteq \bar{\mathcal{C}}\}.$$

Ne segue $\tau(\bar{\mathcal{C}}) = \min_{i=1,\ldots,n_c} \tau(i, \bar{\mathcal{C}})$ dove

$$\tau(i, \bar{\mathcal{C}}) = \left\lfloor \frac{b_i}{\sum_{p \in P} |a_i(p)|} \right\rfloor \tag{3.10}$$

e $\lfloor\ \rfloor$ indica l'operatore floor.

L'enunciato precedente deriva da un risultato presentato in [3] dove viene affrontato il problema di massimizzare il volume degli ipercubi inclusi in politopi.

3.4.2 Determinazione dei punti interni

In questo paragrafo si espone la formulazione di un problema di programmazione lineare intero (LIPP) per ricercare il punto $\boldsymbol{c}$ interno all'insieme $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

L'esigenza di trovare un insieme di marcature decentralizzate oltre che massimale in grado di assicurare buona fairness tra i posti, suggerisce di far coincidere questo punto $\boldsymbol{c}$ con il centro dell'ipercubo intero massimale incluso in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Proposizione 3.4.3. *Si consideri un insieme di marcature legali $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ e si assuma che tale insieme sia limitato almeno in una direzione p_i, $i = 1, \ldots, m$. Questa è la principale condizione richiesta affinchè la soluzione non sia all'infinito. Tale condizione non è un'assunzione limitativa poichè i sistemi che si considerano nella realtà sono sempre limitati fisicamente.*

Il centro $\boldsymbol{c}$ *e il lato* 2τ *dell'ipercubo intero massimale in* $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ *sono ottenuti risolvendo il seguente problema di programmazione lineare intera*

$$\begin{cases} \max\limits_{\boldsymbol{c},\tau} \tau \\ \textit{s.t.} \\ \boldsymbol{w}_i\, \boldsymbol{c} + \sum\limits_{p \in P} |w_i(p)| \tau \leq k_i \quad \forall i = 1 \dots n_c \quad (a) \\ \tau \cdot \mathbf{1_m} \leq \boldsymbol{c} \qquad\qquad\qquad\qquad\quad (b) \\ \tau \in \mathbb{R}_0^+, \quad \boldsymbol{c} \in \mathbb{R}^m \qquad\qquad\qquad (c) \end{cases} \tag{3.11}$$

dove $\mathbf{1_m}$ *è un vettore colonna* m*-dimensionale di elementi pari a uno.*

La risoluzione di un problema di programmazione lineare intera è uno strumento usato in questo lavoro di tesi per giungere a un risultato, ma non costituisce il fulcro su cui focalizzare l'attenzione, per cui non ci si sofferma sulla dimostrazione di tale problema, essendo questa già esposta in [1].

Qualora $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ risultasse illimitato in ogni direzione p_i, $i = 1, \dots, m$ si riscrive l'insieme di marcature legali con l'aggiunta di una condizione che limiti il flusso desiderato in almeno una direzione.

3.4.3 Algoritmi proposti

In [1] sono presentati 3 differenti algoritmi per la ricerca di un box intero in $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$. Solo due dei tre algoritmi proposti sono stati considerati per lo svolgimento di questa tesi e sono stati sviluppati al fine di rendere più veloce possibile l'esecuzione.

Di seguito viene formulato il primo algoritmo proposto.

Algoritmo 3.4.1.

1. Let $\boldsymbol{k}^0 = \tilde{\boldsymbol{k}}$, $U_0 = \{1, \dots, m\}$.

2. For $s = 1$ to m do

2.1. let $\bar{\jmath}_s$ un indice arbitariamente scelto in U_{s-1}

2.2. let $\bar{\imath} = \text{argmin}_{i \in \{1,\dots,n_c\}} \left\lfloor \dfrac{k_i^{s-1}}{|\tilde{w}_i(p_{\bar{\jmath}_s})|} \right\rfloor$

2.3. if $\dfrac{k_{\bar{\imath}}^{s-1}}{\tilde{w}_{\bar{\imath}}(p_{\bar{\jmath}_s})} < 0$, then

$$\text{let } l(p_{\bar{\jmath}_s}) = \left\lfloor \frac{k_{\bar{\imath}}^{s-1}}{\tilde{w}_{\bar{\imath}}(p_{\bar{\jmath}_s})} \right\rfloor,$$

$$\text{let } u(p_{\bar{\jmath}_s}) = \min_{i \in \{1,\dots,n_c\} \,:\, \frac{k_i^{s-1}}{\tilde{w}_i(p_{\bar{\jmath}_s})} \geq 0} \left\lfloor \frac{k_i^{s-1}}{\tilde{w}_i(p_{\bar{\jmath}_s})} \right\rfloor$$

else

$$\text{let } u(p_{\bar{\jmath}_s}) = \left\lfloor \frac{k_{\bar{\imath}}^{s-1}}{\tilde{w}_{\bar{\imath}}(p_{\bar{\jmath}_s})} \right\rfloor,$$

$$\text{let } l(p_{\bar{\jmath}_s}) = \min_{i \in \{1,\dots,n_c\} \,:\, \frac{k_i^{s-1}}{\tilde{w}_i(p_{\bar{\jmath}_s})} < 0} \left\lfloor \frac{k_i^{s-1}}{\tilde{w}_i(p_{\bar{\jmath}_s})} \right\rfloor$$

endif

2.4. for $i = 1$ to n_c do

if $\dfrac{k_i^{s-1}}{\tilde{w}_i(p_{\bar{\jmath}_s})} < 0$, do

let $k_i^s = k_i^{s-1} - l(p_{\bar{\jmath}_s}) w_i(p_{\bar{\jmath}_s})$

else

let $k_i^s = k_i^{s-1} - u(p_{\bar{\jmath}_s}) w_i(p_{\bar{\jmath}_s})$

endif

2.5. let $U_s = U_{s-1} \setminus \{\bar{\jmath}_s\}$. ■

Questo algoritmo è costituito da m passi e assegna a ciascun posto il più piccolo lower bound e il più grande upper bound in modo da garantire il soddisfacimento dei vincoli.

Si procede all'assegnamento dei boundaries nel seguente modo. Allo step 2.1 si sceglie un indice arbitario $\bar{j}_s$ che individua un posto $p_{\bar{\jmath}_s}$ per il quale si cerca il vincolo più restrittivo nella direzione di $p_{\bar{\jmath}_s}$. Se il vincolo più restrittivo corrisponde a un'intersezione negativa con gli assi coordinati, allora si assegna prima il lower bound a $p_{\bar{\jmath}_s}$; l'upper bound dovrebbe essere calcolato valutando solo i vincoli che forniscono un'intersezione positiva con gli assi in direzione $p_{\bar{\jmath}_s}$. Un ragionamento duale si ripete se il vincolo più restrittivo corrisponde a un'intersezione positiva con gli assi coordinati.

Dopo aver valutato il bound per il posto $p_{\bar{j}_s}$, allo step 2.4 avviene l'aggiornamento del lato destro dei vincoli, cioè del termine k_i^s, che dipende dal segno del rapporto $k_i^{s-1}/\tilde{w}_i(p_{\bar{j}_s})$, valutando l'intersezione di ogni vincolo con gli assi nella direzione di $p_{\bar{j}_s}$ usando sia il lower bound, sia l'upper bound.

Questo algoritmo valuta tutti i posti in un ordine arbitrario ed è possibile che i posti scelti per primi abbiano maggiore possibilità di saturare i vincoli; quindi si ha bisogno di cambiare la formulazione dell'algoritmo per poter assicurare una buona fairness tra i posti.

A tale scopo si propone un nuovo algoritmo in grado di calcolare un box intero interno $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*) \subseteq \mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$. Si vedranno sotto quali assunzioni tale algoritmo è in grado di garantire un risultato che sia un box intero interno massimale. Se la massimalità non dovesse essere garantita sarà fatta una modifica che consentirà di verificare anche tale requisito.

L'algoritmo è formulato nel seguente modo.

Algoritmo 3.4.2. [Algoritmo Hypercube Bound]

1. Let $\tau_0 = 0$, $\boldsymbol{W}^0 = \tilde{\boldsymbol{W}}$, $\boldsymbol{k}^0 = \tilde{\boldsymbol{k}}$,
$L_0 = \{1, \ldots, m\}$, $U_0 = \{1, \ldots, m\}$
2. For $s = 1$ to $2m$ do
2.1. let $\mathcal{M}_{s-1} = \mathcal{M}(\boldsymbol{W}^{s-1}, \boldsymbol{k}^{s-1})$
2.2. let $\tau_s = \tau(\mathcal{M}_{s-1})$ (vedere Proposizione (3.4.2))
2.3. let $\bar{\imath}_s, \bar{\jmath}_s$ una coppia di indici arbitariamente scelti in

$$J_s = \left\{ (\bar{\imath}, \bar{\jmath}) \in \mathbb{N} \;\middle|\; s_{\bar{\imath},\bar{\jmath}}^{s-1} = \min_{\substack{j \in L_{s-1} \cup U_{s-1} \\ i \in \{1, \ldots, n_c\}}} s_{i,j}^{s-1} \right\}$$

dove $s_{i,j}^{s-1} = \dfrac{k_i^{s-1} - \tau |\boldsymbol{w}_i^{s-1}| \cdot \mathbf{1}}{|\tilde{w}_i(p_j)|}$
2.4. if $\tilde{w}_{\bar{\imath}_s}(p_{\bar{j}_s}) < 0$, then
let $l_s(p_{\bar{j}_s}) = -\tau_s$
let $L_s = L_{s-1} \setminus \{\bar{\jmath}_s\}$, $U_s = U_{s-1}$
let $neg = 1$, $pos = 0$
else
let $u_s(p_{\bar{j}_s}) = \tau_s$
let $U_s = U_{s-1} \setminus \{\bar{\jmath}_s\}$, $L_s = L_{s-1}$
let $neg = 0$, $pos = 1$

endif

2.5. for $i = 1, \dots, n_c$ do

if $\tilde{w}_i(p_{\bar{j}_s}) < 0$ and $neg = 1$, then

let $k_i^s = k_i^{s-1} + \tau_s \tilde{w}_i(p_{\bar{j}_s})$

let $w_i^s = \begin{cases} 0 & \text{if } j = \bar{j}_s \\ w_i^{s-1}(p_j) & \text{otherwise} \end{cases}$

elseif $\tilde{w}_i(p_{\bar{j}_s}) > 0$ and $pos = 1$, then

let $k_i^s = k_i^{s-1} - \tau_s \tilde{w}_i(p_{\bar{j}_s})$

let $w_i^s = \begin{cases} 0 & \text{if } j = \bar{j}_s \\ w_i^{s-1}(p_j) & \text{otherwise} \end{cases}$

else

let $k_i^s = k_i^{s-1}$

let $w_i^s(p) = w_i^{s-1}(p), \forall p \in P$

endif

3. let $\boldsymbol{l}^* = \boldsymbol{l}_{2m}$, $\boldsymbol{u}^* = \boldsymbol{u}_{2m}$.

■

- L'algoritmo si basa su $2m$ step iterativi e a ogni step viene definita una GMEC $(\boldsymbol{W}^s, \boldsymbol{k}^s)$; allo step iniziale si sceglie $(\boldsymbol{W}^0, \boldsymbol{k}^0) = (\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$. Si indica $\mathcal{M}_s = \mathcal{M}(\boldsymbol{W}^s, \boldsymbol{k}^s)$.

- Allo step s si trova un ipercubo intero interno massimale in $\mathcal{M}_{s-1}$ usando la Proposizione (3.4.2) e indicando con τ_s il corrispondente lato.

- A ogni step si assegna il lower bound o l'upper bound a un posto appartenente al supporto della GMEC corrente, il cui valore coincide con il lato dell'ipercubo corrente. Se allo step s è stato selezionato il posto p_{js}, a seconda del segno di $\tilde{w}_i(p_{\bar{j}_s})$, si avrà $u^*(p_{j_s}) = \tau_s$ oppure $l^*(p_{j_s}) = -\tau_s$.

- La scelta del posto da considerare è essenziale per essere sicuri di ottenere un box interno massimale. Si consideri un vincolo $\tilde{\boldsymbol{w}}_i \cdot \boldsymbol{m} \leq \tilde{k}_i$ e un ipercubo con lato τ che soddisfa tale relazione. Si definisce la *slack* del vincolo i come $s_i = \tilde{k}_i - \tau |\tilde{\boldsymbol{w}}_i| \cdot \mathbf{1}$ dove $\mathbf{1}$ è un vettore colonna di dimensione m i cui elementi sono tutti uno e $|\tilde{\boldsymbol{w}}_i|$ è un vettore riga in cui la j-esima componente per $j = 1, \dots, m$ è uguale a $|\tilde{w}_i(p_j)|$. Dunque si può definire la *slack* relativa al vincolo i rispetto al posto p_j come

$$s_{i,j} = \frac{\tilde{k}_i - \tau |\tilde{\boldsymbol{w}}_i| \cdot \mathbf{1}}{|\tilde{w}_i(p_j)|}. \tag{3.12}$$

Si sceglie di assegnare a ogni step un lower bound o un upper bound a un posto che corrisponde al più piccolo valore della slack relativa che viene indicata con $s_{\bar{\imath}_s,\bar{\jmath}_s}$.

- Si definisce un nuovo insieme di GMEC i cui supporti possono non essere inclusi in quei posti per i quali il limite è già stato assegnato nei precedenti step. Ad esempio, se si assegna un upper bound a $p_{\bar{\jmath}_s}$, nel caso in cui $pos = 1$ e $\tilde{w}_i(p_{\bar{\jmath}_s}) > 0$, allora si elimina $p_{\bar{\jmath}_s}$ dal supporto dell' i-esimo vincolo, altrimenti si mantiene. Se si elimina tale posto, allora i pesi associati ai rimanenti posti non cambiano mentre $\boldsymbol{k}^s$ è aggiornato a $k_i^s = k_i^{s-1} - \tau_s \tilde{w}_i(p_{\bar{\jmath}_s}) \leq k_i^{s-1}$. Se si indica con $U_s \cup L_s$ l'insieme di indici di posti nel supporto di $\mathcal{M}_{s-1}$ al quale un upper bound o un lower bound non è stato assegnato allo step s, possono verificarsi due casi differenti:

  – se $\tilde{w}_{\bar{\imath}_s}(p_{\bar{\jmath}_s}) > 0$ i due vincoli

$$\begin{cases} \sum_{j \in U_s \cup L_s} \tilde{w}_i^s(p_j) m(p_j) \leq k_i^s - \tau_s \tilde{w}_i^s(p_{\bar{\jmath}_s}) \\ m(p_{\bar{\jmath}_s}) \leq \tau_s \end{cases} \tag{3.13}$$

  garantiscono

$$\sum_{j \in U_{s-1} \cup L_{s-1}} \tilde{w}_i^s(p_j) m(p_j) \leq k_i^s; \tag{3.14}$$

  – se $\tilde{w}_{\bar{\imath}_s}(p_{\bar{\jmath}_s}) < 0$ i due vincoli

$$\begin{cases} \sum_{j \in U_s \cup L_s} \tilde{w}_i^s(p_j) m(p_j) \leq k_i^s + \tau_s \tilde{w}_i^s(p_{\bar{\jmath}_s}) \\ m(p_{\bar{\jmath}_s}) \geq -\tau_s \end{cases} \tag{3.15}$$

  garantiscono

$$\sum_{j \in U_{s-1} \cup L_{s-1}} \tilde{w}_i^s(p_j) m(p_j) \leq k_i^s. \tag{3.16}$$

  Le espressioni (3.13) oppure (3.15) garantiscono il soddisfacimento della GMEC al precedente step.

L'implementazione di questo algoritmo verrà presentata nel capitolo successivo.

L'algoritmo così formulato risulta essere più esaustivo e completo rispetto all'Algoritmo (3.4.1) poichè, nonostante il numero di step sia superiore (esattamente

il doppio), ognuno presenta dei criteri specifici per selezionare i parametri di interesse. Infine l'aggiornamento dei vincoli non riguarda solo la costante k_i, ma anche il vettore $\boldsymbol{w}_i$.

Si presenta ora una condizione che, se verificata, indica che l'ipercubo trovato con l'algoritmo appena esposto, è massimale.

Proposizione 3.4.4. *Sia $(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$ un vincolo centralizzato e siano $\boldsymbol{l}^*$ e $\boldsymbol{u}^*$ i vettori finali del lower bound e dell'upper bound calcolati per mezzo dell'Algoritmo HB. Se esiste un indice $\mu \leq 2m$ tale che la sequenza di valori di τ trovati con l'Algoritmo HB soddisfi la condizione*

$$\tau_1 < \tau_2 < \cdots < \tau_\mu = \tau_{\mu+1} = \cdots = \tau_{2m} \tag{3.17}$$

allora $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ è un box interno massimale *incluso in $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$.*

Tale proposizione afferma che se la sequenza di valori di τ è strettamente crescente, con la possibile eccezione che i valori in coda restino costanti, allora si ottiene un box interno massimale. Se ciò non si verificasse, si esegue un algoritmo in grado di garantire che il box risultante sia massimale.

L'algoritmo proposto è il seguente.

Algoritmo 3.4.3. [Algoritmo Hypercube Maximal Bound]

1. Eseguire l'Algoritmo HB. Si assuma che la sequenza di τ sia $\tau_1 \leq \tau_2 \cdots \leq \tau_\mu = \cdots = \tau_{2m}$.
2. Let $\bar{\boldsymbol{l}}^0 = \boldsymbol{l}_{2m}$, $\bar{\boldsymbol{u}}^0 = \boldsymbol{u}_{2m}$
3. For $s = 1$ to $\mu - 1$ do
   - **3.1.** for $j = 1, \ldots, 2m$, $j \neq \bar{\jmath}_s$
     let $\bar{u}^s(p_j) = \bar{u}^{s-1}(p_j)$
     let $\bar{l}^s(p_j) = \bar{l}^{s-1}(p_j)$
   - **3.2.** if $u_{2m}(p_{\bar{\jmath}_s}) = \tau_s$ let $pos = 1$ else let $neg = 1$
   - **3.3.** if pos let

$$\bar{u}^s(p_{\bar{\jmath}_s}) = \begin{cases} \bar{u}^{s-1}(p_{\bar{\jmath}_s}) & \text{if } \tau_s < \tau_{s+1} \\ \bar{u}^{s-1}(p_{\bar{\jmath}_s}) + & \\ \quad + \min\limits_{i \in \{1,\ldots,n_c\}} \left\lfloor \dfrac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{x}}_i^T}{|\tilde{w}_i(p_{\bar{\jmath}_s})|} \right\rfloor & \\ & \text{if } \tau_s = \tau_{s+1} \end{cases}$$

else

$$\bar{l}^s(p_{\bar{j}_s}) = \begin{cases} \bar{l}^{s-1}(p_{\bar{j}_s}) & \text{if } \tau_s < \tau_{s+1} \\ \bar{l}^{s-1}(p_{\bar{j}_s}) + & \\ \quad - \min\limits_{i \in \{1,\dots,n_c\}} \left\lfloor \dfrac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{x}}_i^T}{|\tilde{w}_i(p_{\bar{j}_s})|} \right\rfloor & \\ & \text{if } \tau_s = \tau_{s+1} \end{cases}$$

4. let $\boldsymbol{l}^* = \bar{\boldsymbol{l}}^{\mu-1}$, $\boldsymbol{u}^* = \bar{\boldsymbol{u}}^{\mu-1}$. ■

Questo algoritmo richiede la precedente applicazione dell'Algoritmo HB per poter essere eseguito, poichè richiede la conoscenza della sequenza di valori τ oltre che del lower bound $\boldsymbol{l}$ e dell'upper bound $\boldsymbol{u}$.

Il numero di step di cui si compone l'Algoritmo HMB, pari a $\mu - 1$, è sempre inferiore a $2m$, essendo μ pari al massimo a $2m$. Questo significa che non per tutti i boundaries, lower o upper, ci potrà essere un incremento o un decremento.

La soluzione trovata con l'Algoritmo HB fornisce un box interno massimale quando la sequenza di τ è strettamente crescente, tranne nella coda in cui la sequenza può essere costante, ma non viene garantita la massimalità del box se due o più valori di τ che non si trovano in coda sono uguali. Perciò si cercano tutte le variabili alle quali corrisponde lo stesso upper bound che differisce da τ_m e si verifica se il loro upper bound o lower bound possano essere, rispettivamente aumentato o diminuito ulteriormente. Si aumenta o si diminuisce di una quantità in accordo con i vincoli dati, e si va avanti con l'esplorazione.

Proposizione 3.4.5. *Sia* $(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$ *un vincolo centralizzato. Siano* $\boldsymbol{l}^*$ *e* $\boldsymbol{u}^*$ *i vettori finali di lower bound e upper bound determinati con l'Algoritmo HMB. Allora* $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ *è un* box interno massimale *incluso in* $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$.

Dimostrazione. Se $\tau_s < \tau_{s+1}$ per $s = 1, \dots, \mu-1$ e $\tau_s = \tau_{s+1}$ per $s = \mu, \dots, 2m-1$, allora la soluzione fornita dall'Algoritmo HMB coincide con la soluzione dell'Algoritmo HB e il risultato segue dalla Proposizione (3.4.4).

Sia r il più piccolo valore di $s \in \{1, \dots, \mu - 1\}$ tale che $\tau_s = \tau_{s+1}$. Si supponga che allo step s dell'Algoritmo HMB risulti $u_{2m}(p_{\bar{j}_s}) \neq \tau_s$ e, per esempio, un lower

bound sia assegnato al posto $p_{\bar{\jmath}_s}$. Allora allo step 3.3 si impone

$$\begin{aligned} \bar{l}^r(p_{\bar{\jmath}_r}) &= \bar{l}^{r-1}(p_{\bar{\jmath}_r}) - \min_{i\in\{1,\dots,n_c\}} \left\lfloor \frac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}_i^{s-1}}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} \right\rfloor \\ &= \bar{l}^{r-1}(p_{\bar{\jmath}_r}) - \min_{i\in\{1,\dots,n_c\}} \left\lfloor \frac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}_i^{0}}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} \right\rfloor \end{aligned}$$

mentre $\bar{\boldsymbol{l}}^r(p_j) = \bar{\boldsymbol{l}}^0(p_j)$ per $j \neq \bar{\jmath}_r$.

Sia $\bar{k}_i^0 = k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}_i^0$ e $\bar{k}_i^r = k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}_i^r$, cosicchè

$$\begin{aligned} \bar{k}_i^r &= k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}_i^0 - |\tilde{w}_i(p_{\bar{\jmath}_r})| \min_{i\in\{1,\dots,n_c\}} \left\lfloor \frac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{u}}^0}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} \right\rfloor \\ &= \bar{k}_i^0 - |w_i(p_{\bar{\jmath}_r})| \min_{i\in\{1,\dots,n_c\}} \left\lfloor \frac{\bar{k}_i^0}{|w_i(p_{\bar{\jmath}_r})|} \right\rfloor \end{aligned}$$

e

$$\frac{\bar{k}_i^r}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} = \frac{\bar{k}_i^0}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} - \min_{i\in\{1,\dots,n_c\}} \left\lfloor \frac{\bar{k}_i^0}{|\tilde{w}_i(p_{\bar{\jmath}_r})|} \right\rfloor < 1.$$

Perciò dalla Proposizione (3.4.1), $\mathcal{B}(\bar{\boldsymbol{l}}_r, \bar{\boldsymbol{l}}_r)$ è un box interno massimale. Analogamente, se si indica con q il più piccolo valore di $s \in \{r+1, \dots \mu-1\}$ tale che $\tau_s = \tau_{s+1}$, si dimostra che

$$\frac{\bar{k}_i^q}{|\tilde{w}_i(p_{\bar{\jmath}_q})|} < 1.$$

Rieseguendo iterativamente lo stesso ragionamento fino ad avere considerato tutti i posti, si conclude che $\mathcal{B}(\bar{\boldsymbol{l}}^{\mu-1}, \bar{\boldsymbol{u}}^{\mu-1})$ è un box interno massimale. □

Osservazione 3.4.1. *Nell'Algoritmo HMB si è assunto che upper bound e lower bound siano stati aumentati, quando possibile, seguendo lo stesso ordine con cui sono stati assegnati allo step 2. Tuttavia questa non è la sola soluzione ammissibile. Le variabili che condividono lo stesso upper bound o lower bound possono essere esaminate in qualsiasi ordine e ciò in generale prevede differenti vincoli decentralizzati.*

3.5 Caso B: posti non singleton

Fino ad ora sono stati studiati degli algoritmi che determinano i boundaries di un ipercubo incluso in $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$ e si è poi stabilita una regola che consente di estendere i risultati, senza violare i vincoli, per trovare l'ipercubo massimo.

Ora, a partire da questi risultati, si vuole estendere la trattazione al caso in cui la suddivisione dei posti sia non singleton, ma l'insieme dei posti P sia arbitrariamente partizionato in un sottoinsieme di posti p_i, per trovare l'insieme di GMEC decentralizzate.

A partire dalla definizione di $\boldsymbol{W}^{(i)}$ matrice intera $n_c \times m$ e $\boldsymbol{k}^{(i)}$ vettore intero di dimensione n_c, in [1] si propone una soluzione per calcolare ν GMEC decentralizzate, ognuna definita da n_c vincoli e si sceglie $n_i = n_c$ per tutti i valori di i. Si vuole dimostrare come si possa ottenere una soluzione massimale partendo dalla soluzione dell'Algoritmo HMB.

Definizione 3.5.1. *Sia $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ un box interno massimale in $(\boldsymbol{W}, \boldsymbol{k})$. Secondo la Proposizione (3.2.1) si definisce un insieme di ν GMEC decentralizzate $(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)})$, $i = 1 \dots \nu$, dove $\boldsymbol{W}^{(i)} \in \mathbb{N}^{n_c \times m}$ e $\boldsymbol{k}^{(i)} \in \mathbb{N}^{n_c}$, come segue:*

$$\begin{aligned} \boldsymbol{w}_j^{(i)}(p) &= \begin{cases} \boldsymbol{w}_j(p) & \text{if } p \in P_i \\ 0 & \text{altrimenti} \end{cases} \\ k_j^{(i)} &= \boldsymbol{w}_j^{(i)} \boldsymbol{x}^* \end{aligned} \tag{3.18}$$

$j = 1 \dots n_c$ e per $p \in P$ dove $\boldsymbol{x}^*$ è il punto estremo di $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$(definito come nella Definizione (3.4.2)).

Si noti che $\sum_{i=1}^{\nu} \boldsymbol{w}_j^{(i)} = \boldsymbol{w}_j$ e $k_j^{(i)} = \sum_{i=1}^{\nu} \boldsymbol{w}_j^{(i)} \boldsymbol{x}^* = k_j - s_j$ dove $s_j \geq 0$ è la distanza del vertice massimo del box interno $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ dal j-esimo vincolo.

Ne deriva il seguente risultato.

Proposizione 3.5.1. *Sia $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ un box interno massimale. L'insieme di GMEC decentralizzate $(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)})$ dato dalla Definizione (3.5.1) è un insieme massimale di marcature legali decentralizzate $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.*

Dimostrazione. Si dimostra innanzitutto che $\cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)}) \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Se $\boldsymbol{m} \in \cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)})$, ne deriva che per $j = 1, \dots, n_c$

$$\begin{aligned} &\textstyle\sum_{i=1}^{\nu} \boldsymbol{w}_j^{(i)} \boldsymbol{m} = \boldsymbol{w}_j \boldsymbol{m} \leq \\ &\textstyle\sum_{i=1}^{\nu} k_j^{(i)} = \sum_{i=1}^{\nu} \boldsymbol{w}_j^{(i)} \boldsymbol{x}^* = k_j - s_j > 0. \end{aligned}$$

Allora si conclude che $\boldsymbol{m} \in \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Si assuma che esista un altro insieme di marcature legali decentralizzate $\mathcal{M}^{(i)}$, con $i = 1 \dots \nu$, tale che $\cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)}) \subsetneq \cap_{i=1}^{\nu} \mathcal{M}^{(i)} \subseteq \mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Dunque, $\exists \tilde{\boldsymbol{m}} \in \cap_{i=1}^{\nu} \mathcal{M}^{(i)}$ ma $\tilde{\boldsymbol{m}} \notin \cap_{j=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(j)}, \boldsymbol{k}^{(j)})$.

Poichè $\tilde{\boldsymbol{m}} \notin \cap_{j=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(j)}, \boldsymbol{k}^{(j)})$, segue che

$$\exists \bar{\jmath}, \bar{\imath} \text{ s.t. } \boldsymbol{w}_{\bar{\jmath}}^{(\bar{\imath})} \tilde{\boldsymbol{m}} > k_{\bar{\jmath}}^{(\bar{\imath})} = \boldsymbol{w}_{\bar{\jmath}}^{(\bar{\imath})} \boldsymbol{x}^*,$$

$\exists \bar{\imath}$ s.t. $\tilde{\boldsymbol{m}} \notin \mathcal{M}(\boldsymbol{W}^{(\bar{\imath})}, \boldsymbol{k}^{(\bar{\imath})})$.

Poichè il vettore della marcatura è un vettore intero positivo, può verificarsi una di queste due condizioni:

1. $\exists \tilde{\boldsymbol{m}}' = \boldsymbol{m} + \vec{\varepsilon}_k$ tale che $p_k \in P_{\bar{\imath}}$, $\boldsymbol{m} \in \mathcal{M}(\boldsymbol{W}^{(\bar{\imath})}, \boldsymbol{k}^{(\bar{\imath})})$, $\tilde{\boldsymbol{m}}' \notin \mathcal{M}(\boldsymbol{W}^{(\bar{\imath})}, \boldsymbol{k}^{(\bar{\imath})})$.
   Quindi $\tilde{\boldsymbol{m}}'(p) \geq \boldsymbol{u}^*(p) \;\; \forall p \in P_{\bar{\imath}} \setminus p_k$ e $\tilde{\boldsymbol{m}}'(p_k) \geq \boldsymbol{u}^*(p_k) + 1$.
2. $\exists \tilde{\boldsymbol{m}}' = \boldsymbol{m} - \vec{\varepsilon}_k$ tale che $p_k \in P_{\bar{\imath}}$, $\boldsymbol{m} \in \mathcal{M}(\boldsymbol{W}^{(\bar{\imath})}, \boldsymbol{k}^{(\bar{\imath})})$, $\tilde{\boldsymbol{m}}' \notin \mathcal{M}(\boldsymbol{W}^{(\bar{\imath})}, \boldsymbol{k}^{(\bar{\imath})})$.
   Quindi $\tilde{\boldsymbol{m}}'(p) \leq \boldsymbol{l}^*(p) \;\; \forall p \in P_{\bar{\imath}} \setminus p_k$ e $\tilde{\boldsymbol{m}}'(p_k) \leq \boldsymbol{l}^*(p_k) - 1$.

Senza perdita di generalità si supponga di trovarsi nella prima situazione.

Essendo $\mathcal{M}^{(i)}$ un insieme di marcature legali decentralizzate per quanto riguarda P_i si assume che $\tilde{\boldsymbol{m}}'(p) = \boldsymbol{u}^*(p), \;\; \forall p \notin P_i$ poichè $\boldsymbol{u}^* \in \cap_{i=1}^{\nu} \mathcal{M}(\boldsymbol{W}^{(i)}, \boldsymbol{k}^{(i)}) \subset \cap_{i=1}^{\nu} \mathcal{M}^{(i)}$.

Dunque $\tilde{\boldsymbol{m}}' \gneq \boldsymbol{u}^*$ ma $\boldsymbol{w}_j \tilde{\boldsymbol{m}}' = \boldsymbol{w}_j \boldsymbol{u}^* + \boldsymbol{w}_i(p_k) \leq k_j \forall j$. Questa è una contraddizione, essendo $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ un box interno di massima cardinalità. □

Prima di concludere la discussione e proporre un esempio in grado di far comprendere quanto esposto, si vuole sottolineare l'importanza attribuita alla suddivisione dei posti. Per poter definire l'insieme di GMEC decentralizzate è indispensabile conoscere come i posti p_i siano ripartiti nei diversi insieme P_i: infatti non è sufficiente sapere solo quanti insiemi P_i non singleton vi sono, ma anche quali posti p_i vi appartengono. L'informazione legata alla suddivisione dei posti è fondamentale per la determinazione delle GMEC decentralizzate poichè, secondo la relazione (3.18), stabilisce quali sono i pesi della matrice $\boldsymbol{W}$ e del vettore $\boldsymbol{k}$.

Si veda di seguito un esempio per comprendere meglio quanto esposto sopra.

Esempio 3.5.1. *Sia* $\mathcal{M}(\boldsymbol{w}, k) = \{\boldsymbol{m} \mid 3\boldsymbol{m}(p_1) + \boldsymbol{m}(p_2) + \boldsymbol{m}(p_3) \leq 5\}$ *l'insieme di marcature legali. Si assuma che* $P_1 = \{p_1, p_2\}$ *e* $P_2 = \{p_3\}$.

Si applica l'Algoritmo HB per ottenere un box interno massimale in $\mathcal{M}(\boldsymbol{w}, k)$ *e si ottengono i vettori che rappresentano rispettivamente lower bound e upper bound* $\boldsymbol{l}^* = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ *e* $\boldsymbol{u}^* = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$.

Infine dalla Definizione (3.5.1) si ottiene

$$\mathcal{M}(\boldsymbol{w}^{(1)}, k^{(1)}) = \{\boldsymbol{m} \in \mathbb{N}^2 \mid 3\boldsymbol{m}(p_1) + \boldsymbol{m}(p_2) \leq 4\},$$
$$\mathcal{M}(\boldsymbol{w}^{(2)}, k^{(2)}) = \{\boldsymbol{m} \in \mathbb{N} \mid \boldsymbol{m}(p_3) \leq 1\}.$$

■

Capitolo 4

Software sviluppato

Attraverso la risoluzione di un problema di programmazione lineare intera si è realizzato un passo fondamentale per la ricerca del centro e del lato, e di conseguenza, dei boundaries dell'ipercubo intero massimale. Si cercherà di spiegare come è stato risolto tale problema, senza entrare nei dettagli della tecnica di risoluzione, poichè essa non costituisce argomento di tale lavoro, ma solo un modo per arrivare alla soluzione.

Successivamente si farà un breve cenno sul software utilizzato, che ha consentito di implementare gli algoritmi risolutivi, e verranno introdotte le funzioni create. Per ognuna di queste verranno definiti gli argomenti in ingresso e ciò che si vuole determinare a conclusione dell'esecuzione della funzione, cioè cosa fornisce in uscita. Inoltre si spiegherà il significato di ogni funzione chiarendo le funzionalità implementate.

4.1 Linear Integer Programming Problem

Un problema di programmazione lineare intero (LIPP) [4] è un problema che ricerca il valore minimo o massimo di una funzione lineare in presenza di vincoli lineari dati in termini di uguaglianza o disuguaglianza. Ciò che differenzia tale problema dai problemi di programmazione lineare generici è il vincolo che le variabili possano assumere solo valori interi. Un problema di questo tipo può

essere formulato, in termini matriciali, nel seguente modo:

$$\begin{cases} \max & \boldsymbol{cx} \\ \text{s.t.} & \boldsymbol{Ax} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{N}^n, \end{cases} \tag{4.1}$$

in cui $\boldsymbol{A}$ è una matrice a rango pieno $m \times n$, $\boldsymbol{x}$ è il vettore colonna n-dimensionale delle variabili, $\boldsymbol{b}$ è il vettore colonna m-dimensionale dei termini noti, $\boldsymbol{c}$ è il vettore riga n-dimensionale dei coefficienti della funzione obiettivo.

Il problema che si vuole risolvere con la programmazione lineare intera è rappresentato dal sistema (3.11) e viene utilizzato per la ricerca del centro $\boldsymbol{c}$ e del lato 2τ dell'ipercubo intero massimale. Il significato fisico del problema è stato esposto nel capitolo precedente; ora si studierà la formulazione del problema per la ricerca della soluzione.

Ogni problema di programmazione lineare è costituito da tre parti:

1. una funzione obiettivo da massimizzare o minimizzare, in questo caso si vuole massimizare τ;
2. un insieme di vincoli che definiscono lo spazio in cui ricercare il valore ottimo della funzione obiettivo;
3. i vincoli sulle variabili: nel caso in questione, trattandosi di un problema di programmazione lineare intero, le variabili devono essere intere.

Per poter formulare il problema e trovare una soluzione è necessario disporre di tutte queste informazioni.

In questo caso la funzione obiettivo consente di trovare il centro $\boldsymbol{c}$ e il lato 2τ dell'iperbubo intero massimale e dunque, queste saranno le variabili del problema.

I vincoli rappresentati genericamente da $\boldsymbol{Ax} \leq \boldsymbol{b}$ sono costituiti da:

$$\boldsymbol{w}_i \, \boldsymbol{c} + \sum_{p \in P} |w_i(p)| \tau \leq k_i \quad \forall i = 1 \ldots n_c$$

e da:

$$\tau \cdot \mathbf{1_m} \leq \boldsymbol{c}.$$

Poichè si conoscono le variabili del problema (il vettore $\boldsymbol{x}$) e i valori k_i che costituiscono il vettore $\boldsymbol{b}$, resta da determinare la matrice $\boldsymbol{A}$ e, per adempiere a tale compito, si è creata la funzione *trova_A*. Tale funzione prende in ingresso una matrice *W_in*, la quale ha un numero di righe q pari al numero di GMEC singole considerate e un numero di colonne m pari al numero totale di posti considerati.

La funzione *trova_A* calcola la matrice $\boldsymbol{A}$ di dimensione $q \times (m+1)$ costituita dalla concatenazione orizzontale della matrice *W_in* in ingresso e di un vettore $\boldsymbol{F}$ che rappresenta la sommatoria del prodotto dei valori assoluti della riga i-esima della matrice *W_in* per un vettore colonna di dimensione m i cui elementi sono tutti pari a uno. In altre parole, il vettore $\boldsymbol{F}$ rappresenta la somma dei valori assoluti dei termini della i-esima riga della matrice *W_in*.

La moltiplicazione tra la matrice $\boldsymbol{A}$ così determinata e il vettore colonna $\boldsymbol{x}$ delle variabili, formato da $m+1$ elementi (le prime m componenti sono le coordinate del centro $\boldsymbol{c}$, l'ultima componente è la variabile τ), costituisce i vincoli entro i quali cercare la soluzione ottima.

4.1.1 La libreria *glpkmex.lib*

Esistono diversi software che consentono di risolvere un problema come quello appena esposto, ma si è scelto di utilizzare una libreria di Matlab chiamata *glpkmex.lib* poichè questa permette di risolvere problemi di programmazione lineare anche piuttosto complicati e con un numero elevato di variabili, senza doverle enumerare tutte e senza dover riscrivere tutti i vincoli singolarmente. Infatti l'omonima funzione contenuta al suo interno consente di risolvere problemi di programmazione lineare che hanno la stessa struttura del problema descritto in (3.11), conoscendo la matrice $\boldsymbol{A}$, il vettore $\boldsymbol{x}$ e i vettori $\boldsymbol{b}$ e $\boldsymbol{c}$.

La sintassi della chiamata è:

```
[X_opt,F_opt,STATUS,EXTRA]=glpkmex
(SENSE,C,A,B,CTYPE,LB,UB,VARTYPE,PARAM,LPSOLVER,SAVE).
```

Di seguito viene spiegato ogni campo.

SENSE: indica di che tipo di problema si tratta, se di minimizzazione (SENSE=1), o di massimizzazione (SENSE=-1);

C: è un vettore colonna contenente i coefficienti della funzione obiettivo;

A: è la matrice dei vincoli, trovata con la funzione *trova_A*;

B: è un vettore colonna contenente i termini noti di ogni vincolo: ciò che si trova alla destra del segno di uguaglianza o di disuguaglianza;

CTYPE: è un vettore colonna contenente la lettera che codifica il segno di ogni vincolo: 'F' variabile libera (illimitata), 'U' variabile con limite superiore, 'S' variabile fissa, 'L' variabile con limite inferiore, 'D' variabile a doppio limite;

LB: è un vettore colonna che contiene il lower bound di ogni variabile;

UB: è un vettore colonna contenente l'upper bound di ogni variabile;

VARTYPE: è un vettore colonna che contiene il tipo di variabili usate, continue ('C') o intere ('I');

LPSOLVER: permette di decidere il metodo risolutivo tra il metodo del simplesso (LPSOLVER=1) o il metodo del punto interno (LPSOLVER=2);

In questo caso, non è necessario specificare i parametri PARAM e SAVE. In uscita è richiesto di conoscere solo *X_opt* e *F_opt*: il primo termine è un vettore con $m+1$ componenti, in cui le prime m sono le coordinate del centro $\boldsymbol{c}$ e l'ultima è il lato τ; il secondo termine è il valore massimo della funzione obiettivo, cioè il valore massimo del lato τ: infatti tale valore è sempre uguale all'ultima componente del vettore *X_opt*.

4.1.2 Esempio di risoluzione

Sia dato il seguente insieme di GMEC:

$$\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{ \quad \boldsymbol{m} \in \mathbb{N}^2 \ \mid \ m(p_1) + 2m(p_2) \leq 8, \\ -2m(p_1) + m(p_2) \leq 0\}.$$

Data la matrice $W\_in = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$, la matrice $\boldsymbol{A}$ calcolata con la funzione *trova_A* è pari a $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 1 & 3 \end{bmatrix}$, e il problema di programmazione lineare intero è così formulato:

- SENSE=-1, poichè si vuole massimizzare la funzione obiettivo;
- C=$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ è un vettore colonna con 0 nelle prime m posizioni e 1 nell'ultima riga;
- $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 1 & 3 \end{bmatrix}$ è la matrice 2×3 trovata con la funzione *trova_A*;
- B=$\begin{bmatrix} 8 \\ 0 \end{bmatrix}$ è il vettore colonna che contiene i coefficienti k_i;
- CTYPE=$\begin{bmatrix} 'U' \\ 'U' \end{bmatrix}$ è un vettore colonna che contiene un numero di componenti pari al numero di righe della matrice $\boldsymbol{A}$. In questo caso i vincoli sono tutti rappresentati da disequazioni dello stesso tipo e sono codificati con la lettera U;
- LB=$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ è un vettore colonna di dimensione pari al numero di variabili che contiene i limiti inferiori che esse possono assumere;
- UB=$\begin{bmatrix} Inf \\ Inf \\ Inf \end{bmatrix}$ è un vettore colonna di dimensione pari al numero di variabili che contiene i limiti superiori che esse possono assumere;
- VARTYPE=$\begin{bmatrix} 'C' \\ 'C' \\ 'C' \end{bmatrix}$ è un vettore colonna di dimensione pari al numero di variabili e permette di decidere se usare variabili intere (I) o continue (C);
- LPSOLVER=1 impone di cercare la soluzione usando il metodo del simplesso.

A questo punto tutti i parametri che descrivono il problema sono stati definiti ed è possibile richiamare la funzione *glpkmex*, con la sintassi espressa sopra, nella

Command Window di Matlab e trovare subito la soluzione che fornisce *X_opt* $= \begin{bmatrix} 2.6 & 0 & 1.7 \end{bmatrix}$ e *F_opt* $= 1.7$.

Prima di concludere si osservi che nel definire il parametro VARTYPE si è scelto di usare variabili continue ('C'), proprio come definito nel sistema (3.11) che imponeva τ e $\boldsymbol{c}$ appartenere all'insieme dei numeri reali; ciò però va in contraddizione con il vincolo di variabili intere che definiscono il problema.
Se si fossero scelte variabili intere ponendo VARTYPE= $\begin{bmatrix} 'I' \\ 'I' \\ 'I' \end{bmatrix}$, si sarebbe ottenuto *X_opt* $= \begin{bmatrix} 2 & 1 & 1 \end{bmatrix}$ e *F_opt* $= 1$.

E' stata tale questione a suggerire un'interpretazione alternativa del problema.

4.1.3 Un risultato nel dominio intero

Nel sistema (3.11) quando sono stati specificati i vincoli sulle variabili si è imposto:

- $\tau \in \mathbb{R}_0^+$,
- $\boldsymbol{c} \in \mathbb{R}^m$.

Durante la risoluzione del LIPP però è emerso che effettivamente tali condizioni non corrispondevano a quanto si desidera applicare. Infatti, τ rappresenta metà lato dell'ipercubo intero massimale, quindi tale valore può assumere solo valori dettati dalla condizione che 2τ sia intero. Dunque sarà necessario, nel corso della risoluzione del LIPP, imporre la condizione che τ appartenga all'insieme di numeri interi non negativi ovvero, traducendo tale frase in espressione matematica, si ha: $\tau \in \mathbb{Z}_0^+$.

Anche per quel che riguarda il centro $\boldsymbol{c}$ si possono assumere due nuove condizioni che possono essere espresse come segue:

1. si considera $\boldsymbol{c}$ appartenente all'insieme dei numeri interi, ovvero $\boldsymbol{c} \in \mathbb{Z}^m$;
2. si considera $\boldsymbol{c}$ appartenente all'insieme dei numeri reali, ma allora bisognerà aggiungere un'ulteriore condizione, dopo aver riportato nel sistema iniziale

i boundaries trovati con gli algoritmi proposti relativi al sistema traslato, per avere una soluzione intera poichè la traslazione degli assi comporterebbe dei boundaries non interi.

La prima condizione risulta innovativa perchè, a differenza di quanto espresso nel sistema (3.11) in cui si assumevano valori reali per il centro, in questo caso si richiede che il centro abbia coordinate intere. Questa idea è sicuramente la più intuitiva perchè non aggiunge nessun ulteriore passaggio agli algoritmi risolutivi.

Ciò che risulta essere piuttosto interessante è la seconda idea proposta sulla quale ci si sofferma di più poichè rappresenta una condizione aggiuntiva alla determinazione dei boundaries che fino ad ora erano stati calcolati usando solo coordinate reali per il centro $\boldsymbol{c}$. Quest'ulteriore condizione da imporre al termine dell'algoritmo permette, dopo aver riportato il problema al sistema di coordinate originali, di trovare ancora un ipercubo intero. Per verificare tale condizione si impone quindi un 'restringimento' dell'ipercubo: infatti nel sistema traslato si ottengono un lower bound e un upper bound costituiti da coordinate intere. Durante la traslazione tale condizione non sarà più garantita se il centro $\boldsymbol{c}$ ha coordinate reali. Ciò può essere osservato considerando la condizione $\boldsymbol{m} = \boldsymbol{m}^{'} + \boldsymbol{c}$ che trasla l'origine di un nuovo sistema nel punto $\boldsymbol{c}$. Questo significa che lower bound e upper bound nel sistema originale non avranno più coordinate intere, andando contro l'ipotesi iniziale di coordinate intere dell'ipercubo.

Tale problema può essere risolto nel seguente modo:

Una volta trovati lower bound e upper bound per il sistema traslato, si trovano lower bound e upper bound $\boldsymbol{i}$*nteri per il sistema originale con due operazioni:*

- *per il lower bound, si applica l'operatore floor a ciascun valore che costituisce il bound e si somma 1, in modo che tale valore si trovi ancora all'interno dell'ipercubo intero trovato nel sistema traslato;*
- *per l'upper bound, si applica il solo operatore floor ai valori che costituiscono il bound.*

In questo modo si ha la sicurezza di:

1. garantire al sistema che le coordinate siano intere;
2. avere coordinate maggiori, o al più uguali, di quelle dell'ipercubo trovato nel sistema traslato nel caso dei lower bound e coordinate minori, o al

più uguali, di quelle dell'ipercubo del sistema traslato nel caso dell'upper bound.

Prima di concludere si osservi che l'ipercubo trovato nel sistema di coordinate originali risulterà più piccolo rispetto a quello trovato nel sistema traslato, dove con l'espressione 'più piccolo' si intende che il nuovo ipercubo avrà un'area inferiore se il problema ha due dimensioni, ha un volume inferiore se si hanno tre dimensioni, ecc. rispetto all'ipercubo intero.

Esempio 4.1.1. *Si supponga che dopo la traslazione al sistema di coordinate originale si ottengano un lower bound pari a* $\boldsymbol{l} = \begin{bmatrix} 1.3 & 1.1 \end{bmatrix}$ *e un upper bound pari a* $\boldsymbol{u} = \begin{bmatrix} 2.7 & 3.2 \end{bmatrix}$*, allora nel sistema originale tali limiti avranno i seguenti valori:*

- $l(p_1) = \lfloor 1.3 \rfloor + 1 = 1 + 1 = 2$ *rappresenta la prima coordinata del lower bound;*
- $l(p_2) = \lfloor 1.1 \rfloor + 1 = 1+1 = 2$ *rappresenta la seconda coordinata del lower bound;*
- $u(p_1) = \lfloor 2.7 \rfloor = 2$ *rappresenta la prima coordinata dell'upper bound;*
- $u(p_2) = \lfloor 3.2 \rfloor = 3$ *rappresenta la seconda coordinata dell'upper bound.*

Come si può notare, l'ipercubo trovato nel sistema di coordinate originali è inferiore a quello trovato nel sistema traslato. Questo fatto comporta una differenza di risultati nei due sistemi, i quali invece devono trovare un ipercubo con le stesse caratteristiche.

Per questa ragione nel seguito di questa tesi si assume che il centro $\boldsymbol{c}$ e il lato τ appartengano all'insieme $\mathbb{Z}$ dei numeri interi.

4.2 L'implementazione degli algoritmi

Matlab è un ambiente di sviluppo che consente di eseguire operazioni usando matrici e vettori, definire nuove variabili associando un nome a un'espressione, creare funzioni complesse combinando funzioni di base e implementare algoritmi. Grazie a una sintassi e una serie di funzioni che agevolano le operazioni matriciali,

Matlab si presta molto bene a tali operazioni: questo rappresenta il principale motivo per cui è stato scelto per implementare gli algoritmi affrontati in questo lavoro.

I file di Matlab sono sostanzialmente di due tipi: script e funzioni. Il primo tipo consente, inizializzando le variabili da usare, di richiamare al suo interno delle funzioni precedentemente create o di eseguire operazioni matriciali. Uno script può essere eseguito richiamando il nome del file nella Command Window. Il secondo permette di creare una funzione con la seguente sintassi:

[argomenti_in_uscita]=nome_funzione(argomenti_in_ingresso).

Tali funzioni possono raggruppare al loro interno un insieme di funzioni più semplici già presenti all'interno dell'ambiente Matlab, che consentono di eseguire le principali operazioni.

4.2.1 La tecnica impiegata

In questa sezione viene descritto lo studio degli algoritmi che precede la loro implementazione al fine di spiegare le motivazioni alla base delle scelte effettuate. Si cerca di creare un legame tra la formulazione degli algoritmi in termini matematici e la loro struttura implementativa.

In questa tesi sono stati implementati l'Algoritmo HB e l'Algoritmo HMB, oltre l'analisi dettagliata di questi, si esegue un ulteriore studio, anch'esso oggetto di implementazione, rivolto al caso in cui la suddivisione dei posti sia di tipo non singleton.

L'Algoritmo HB è il primo algoritmo implementato e per primo determina i valori dei boundaries. Per calcolare, o eventualmente per accertare, i valori massimali di tali boundaries è stato formulato l'Algoritmo HMB, che quindi può considerarsi un seguito dell'Algoritmo HB. Infine, l'ulteriore condizione riguardante la suddivisione dei posti può essere vista come un'estensione dell'Algoritmo HMB. Infatti i risultati dell'Algoritmo HB costituiscono i dati in ingresso all'Algoritmo HMB, i cui risultati vengono elaborati nel caso di una partizione di posti di tipo non singleton. Questo aspetto di successione rappresenta lo schema su cui si basa l'implementazione.

La tecnica adottata per l'implementazione è la seguente. Per ogni step che costituisce l'algoritmo si valuta se esso richiede il calcolo di un parametro per mezzo

di un'espressione algebrica o la verifica di una condizione su un parametro precedentemente calcolato per eseguire l'assegnamento a una o più variabili. A seconda delle richieste, sarà diverso il peso computazionale associato a ciascuno step. In generale si creano delle funzioni che calcolano i parametri richiesti dai diversi step o su cui verranno fatte delle verifiche per l'assegnamento dei valori alle variabili. Quindi l'esecuzione degli algoritmi è costituita da una successione di operazioni che vengono tradotte in una successione di funzioni in cui ognuna calcola un parametro.

Infine, è bene specificare che in alcuni step si richiede di calcolare dei parametri per i quali sono necessari dei dati in ingresso che non sono ancora stati calcolati agli step precedenti. Quindi pensare a una relazione univoca tra il numero di step che compongono gli algoritmi e il numero di funzioni richiamate nel programma non è corretto, poichè ogni step ingloba più funzioni che ricercano diversi valori per poter valutare adeguatamente i parametri utilizzati per ricercare i boundaries.

4.3 Implementazione dell'Algoritmo HB

Per questo algoritmo si è usato uno script, chiamato *algoritmo_HB* in cui è stata definita la matrice *W_in* e il vettore *k_in* che definiscono la GMEC globale, il vettore *X_opt* che rappresenta la soluzione del problema di programmazione lineare intero, la matrice $\mathbf{LU}$ i cui elementi sono inizialmente tutti nulli e due insiemi L e U. Si ha bisogno di creare più funzioni, richiamate all'interno dello script, poichè ogni funzione consente di calcolare un parametro che viene espresso come combinazione di somme o altre operazioni algebriche e di funzioni elementari di Matlab.

La matrice *W_in* e il vettore *k_in*, insieme al vettore *X_opt*, vengono elaborati dalla funzione *data_input* al fine di determinare la matrice $\boldsymbol{W}$ e il vettore $\boldsymbol{k}$ che rappresentano una GMEC traslata, il cui centro è nel punto *X_opt*, e che costituiscono i dati in ingresso alle funzioni richiamate nello script.

La sintassi della funzione *data_input* è:

[k,W]=data_input(K_in,W_in,X_opt).

Gli argomenti in ingresso sono:

- la matrice *W_in* e il vettore *k_in*, determinati dalla GMEC globale;

- *X_opt*, vettore determinato risolvendo un problema di programmazione lineare.

Gli argomenti in uscita sono:

- la matrice $\boldsymbol{W}$ e il vettore $\boldsymbol{k}$, che definiscono una GMEC traslata con origine nel punto *X_opt*.

La matrice $\boldsymbol{W}$ è così determinata:

- le prime righe sono costituite dai vettori $\boldsymbol{w}$ delle GMEC singole;
- le successive righe rappresentano i vincoli aggiuntivi di non negatività indicati da $\boldsymbol{m}' \leq \boldsymbol{c}$.

Essa ha quindi un numero di righe pari alla somma del numero di GMEC singole considerate e del numero di posti presenti nella GMEC multipla. Le prime q righe saranno uguali a quelle delle GMEC, mentre le ultime m righe saranno costituite da una matrice diagonale i cui elementi sono unitari negativi.

Il vettore colonna $\boldsymbol{k}$, il quale è dato dalla concatenazione verticale di un vettore costituito da elementi determinati con l'espressione $k\_in - W\_in \cdot X\_opt$ e del vettore *X_opt* privato dell'ultima riga, ha un numero di righe pari al numero di righe della matrice $\boldsymbol{W}$.

In questo script sono definiti anche la matrice $\boldsymbol{A}$ uguale alla matrice $\boldsymbol{W}$ e il vettore $\boldsymbol{b}$ equivalente al vettore $\boldsymbol{k}$. In particolare per l'Algoritmo HB sono state create le seguenti funzioni:

- *calcola_tau(A,b)*, per trovare il valore τ_s;
- *calcola_S(W,k,tau_s)*, per calcolare la matrice $\boldsymbol{S}$ definita dall'insieme di elementi calcolati come nella relazione (3.12);
- *calcola_IND(S)*, per trovare una coppia di indici $(\bar{i_s}, \bar{j_s})$ associati all'elemento minimo della matrice $\boldsymbol{S}$;
- *calcola_LU(W,tau_s,IND,LU)*, per determinare la matrice LU dei boundaries;

- *calcola_W_k(W,k,tau_s,IND,NEG,POS)*, per calcolare i nuovi valori di $\boldsymbol{W}$ e $\boldsymbol{k}$;

Tale algoritmo viene risolto in un numero di passi pari a $2m$, dove m indica il numero di posti presenti nella GMEC globale.

4.3.1 Calcolo di τ_s

Il primo termine da calcolare, indicato allo step 2.2 dell'algoritmo, richiede il valore di τ_s come definito nell'espressione (3.10), in cui il pedice s indica lo step che si sta eseguendo; in totale, al termine dell'applicazione dell'algoritmo si hanno $2m$ valori di τ. La funzione *calcola_taus* ha la seguente sintassi:

[tau_s]=calcola_tau_s(A,b).

Gli argomenti in ingresso sono:

- $\boldsymbol{A}$, matrice di n_c righe e m colonne, definita uguale alla matrice $\boldsymbol{W}$;
- $\boldsymbol{b}$, vettore colonna di n_c righe, definito uguale al vettore $\boldsymbol{k}$.

Questi termini permettono di definire un generico insieme convesso $\mathcal{C}(\boldsymbol{A}, \boldsymbol{b})$ contenente l'origine.

L'argomento in uscita è:

- τ_s, scalare che rappresenta il lato dell'ipercubo.

Con questa funzione vengono calcolati tanti valori τ quante sono le n_c righe della matrice $\boldsymbol{A}$, poichè ogni valore τ è determinato come il rapporto tra l'i-esima componente del vettore $\boldsymbol{b}$ e la somma dei valori assoluti degli elementi della riga i-esima della matrice $\boldsymbol{A}$, per l'indice i che varia da 1 a n_c. A tutti questi n_c valori di τ si applica l'operatore *floor* che determina la parte intera del valore trovato. Tra tutti gli n_c valori interi di τ così ottenuti, si indica con τ_s quello più piccolo.

Esempio 4.3.1. *sia* $\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$ *e* $\boldsymbol{b} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$.
Si determini τ_s *come indicato nella Proposizione (3.4.2).*

Con la funzione calcola_taus *si ottiene:*

$$\tau_s = min\left\{ \left\lfloor \frac{4}{3} \right\rfloor, \left\lfloor \frac{3}{3} \right\rfloor, \left\lfloor \frac{2}{1} \right\rfloor, \left\lfloor \frac{1}{1} \right\rfloor \right\}$$

percui $\tau_s = 1$.

4.3.2 Calcolo della matrice S

Lo step 2.3 dell'algoritmo richiede di trovare una coppia di indici $(\bar{i}_s, \bar{j}_s)$ associati all'elemento minimo della matrice $\boldsymbol{S}$, i quali potranno essere determinati solo dopo che si conoscono tutti i termini $s_{i,j}$ come definiti in (3.12). Durante lo studio dell'algoritmo si è pensato che potesse essere più semplice considerare i termini $s_{i,j}$ come elementi di una matrice di dimensione uguale alla matrice $\boldsymbol{W}$.

La funzione ha la seguente sintassi:

[S]=calcola_S(W,k,tau_s).

Gli argomenti in ingresso sono:

- $\boldsymbol{W}$, matrice di n_c righe e m colonne;
- $\boldsymbol{k}$, vettore colonna di n_c righe;
- τ_s, scalare determinato con la funzione *calcola taus(A,b)*.

L'unico argomento in uscita è:

- $\boldsymbol{S}$, matrice $n_c \times m$ i cui elementi sono determinati da

$$s_{i,j}^{s-1} = \frac{k_i^{s-1} - \tau |\boldsymbol{w}_i^{s-1}| \cdot \mathbf{1}}{|\tilde{w}_i(p_j)|}. \tag{4.2}$$

Si vuole precisare che si è indicato con τ il valore τ_s trovato con l'apposita funzione.

Per poter costruire la matrice $\boldsymbol{S}$ c'è bisogno di determinare prima di tutto il valore assoluto della i-esima riga della matrice $\boldsymbol{W}$, per i che varia da 1 a n_c. La riga contenente i valori assoluti viene indicata con $\boldsymbol{w}$. Si indica con w_{ij} il valore assoluto dell'elemento della matrice $\boldsymbol{W}$ che si trova alla riga i e alla colonna j.

Ogni elemento componente la matrice $\boldsymbol{S}$ viene calcolato con l'espressione (4.2), come rapporto tra un numeratore in cui vi è la differenza tra uno scalare k_i e uno scalare determinato dal prodotto tra τ_s per un prodotto vettoriale tra il vettore riga $\boldsymbol{w}$ e il vettore colonna $\mathbf{1}$ e un denominatore in cui compare il solo termine w_{ij}.

Esempio 4.3.2. *Sia* $\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$, $\boldsymbol{k} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$ *e* $\tau_s = 1$.
Si determini la matrice $\boldsymbol{S}$.

Ogni elemento di questa matrice sarà dato dall'espressione (4.2).

Con la funzione calcola_S *si trova la seguente matrice* $\boldsymbol{S} = \begin{bmatrix} 1 & 0.5 \\ 0 & 0 \\ 1 & Inf \\ NaN & 0 \end{bmatrix}$.

4.3.3 Ricerca degli indici $(\bar{i_s}, \bar{j_s})$

Nota la matrice $\boldsymbol{S}$, è possibile determinare la coppia di indici $(\bar{i_s}, \bar{j_s})$ di un elemento minimo della matrice in ingresso. La sintassi di questa funzione è:

[IND]=calcola_IND(S).

L'unico elemento in ingresso è:

- $\boldsymbol{S}$, matrice di dimensione $n_c \times m$ determinata con la funzione *calcola_S*.

L'argomento in uscita è:

- **IND**, vettore contenente una coppia di indici corrispondenti a un elemento con valore minimo di $\boldsymbol{S}$.

Questa funzione non contiene nessuna operazione algebrica al suo interno, ma si basa su un semplice confronto di elementi al fine di trovare quello minimo. Infatti, con la funzione base di Matlab min è possibile determinare l'elemento minimo di ciascuna riga. Se la funzione min viene applicata una seconda volta, come è stato fatto, dopo avere trovato l'elemento minimo della riga, trova poi l'elemento minimo tra tutte le righe. *minimo = min(min(S))* assegna a *minimo* il valore più piccolo presente all'interno della matrice $\boldsymbol{S}$ e si vogliono determinare gli indici di riga e di colonna di tale valore. Ciò può essere fatto grazie alla funzione base *find* che ricerca tra tutti gli elementi che costituiscono la matrice $\boldsymbol{S}$ quelli uguali a *minimo* e restituisce i loro indici in due vettori: il primo contiene gli indici di riga, il secondo contiene gli indici di colonna.

E' possibile che il valore *minimo* sia presente più di una volta nella matrice $\boldsymbol{S}$, perciò si è scelto di creare una matrice $\boldsymbol{J}$ come la concatenazione del vettore I degli indici di riga e del vettore E degli indici di colonna entrambi restituiti dalla funzione *find*.

Poichè non vi è un criterio per scegliere una coppia di indici $(\bar{i_s}, \bar{j_s})$, si prendono tali indici uguali a quelli indicati nella prima riga della matrice $\boldsymbol{J}$, così che $\bar{i_s} = \boldsymbol{J}(1,1)$ è uguale all'elemento della matrice $\boldsymbol{J}$ nella prima riga e nella prima colonna e $\bar{j_s} = \boldsymbol{J}(1,2)$ è uguale all'elemento della matrice $\boldsymbol{J}$ nella prima riga e nella seconda colonna.

Esempio 4.3.3. *Sia data la matrice* $\boldsymbol{S} = \begin{bmatrix} 1 & 0.5 \\ 0 & 0 \\ 1 & Inf \\ NaN & 0 \end{bmatrix}$.
Determinare gli indici del valore minimo.

A tale scopo si usa la funzione calcola_IND*, che come prima operazione ricerca il valore minimo tra gli elementi della matrice. Si ottiene* $minimo = 0$*. Poi si creano due vettori: il vettore I degli indici di riga di tutti gli elementi uguali a minimo, e il vettore E degli indici di colonna degli elementi uguali a minimo. Per cui si ha* $\boldsymbol{I} = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$ *e* $\boldsymbol{E} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$*. Si può ora determinare la matrice*

$\boldsymbol{J}$ *affiancando questi due vettori, per cui* $\boldsymbol{J} = \begin{bmatrix} 2 & 1 \\ 2 & 2 \\ 4 & 2 \end{bmatrix}$. *Tra tutte le coppie di indici che costituiscono la matrice* $\boldsymbol{J}$ *si prende una sola coppia di indici e per convenzione si è deciso di prendere sempre gli indici nella prima riga di* $\boldsymbol{J}$. *Quindi* $\mathbf{IND} = \begin{bmatrix} 2 & 1 \end{bmatrix}$.

4.3.4 Determinazione dei boundaries

Una delle funzioni chiave che è stata sviluppata per la risoluzione degli algoritmi è *calcola_LU*. Infatti questa funzione consente di determinare i boundaries che costituiscono il fine degli algoritmi studiati.

La funzione ha la seguente sintassi:

[LU_out,NEG,POS,L_out,U_out]=calcola_LU(W,tau_s,IND,LU,L,U).

Alcuni elementi in ingresso sono già stati menzionati per altre funzioni, come $\boldsymbol{W}$, o trovati creando apposite funzioni, come τ_s o $\mathbf{IND}$, quindi ci si sofferma solo su quegli argomenti che ancora non sono stati considerati:

- $\mathbf{LU}$, matrice $2 \times m$ i cui elementi sono inizialmente tutti nulli. Tale matrice è definita nello script *algoritmo_HB*;
- L, vettore riga i cui elementi sono gli indici dei posti m. Questo insieme è definito nello script *algoritmo_HB*;
- U, vettore riga i cui elementi sono gli indici dei posti m. Anche questo insieme è definito nello script *algoritmo_HB*;

Gli argomenti in uscita invece sono:

- *LU_out*, matrice $2 \times m$ inizialmente costituita da elementi nulli, ciascuno determinato a ogni step dell'algoritmo. Ciascun elemento costituisce un bound relativo al posto indicato dalla colonna. Nella prima riga ci sono i lower bounds, nella seconda riga ci sono gli upper bounds;

- NEG, variabile booleana determinata dal segno dell'elemento $(\bar{i_s}, \bar{j_s})$ della matrice $\boldsymbol{W}$. Se questo termine è minore di 0, allora NEG=1, altrimenti NEG=0;
- POS, variabile booleana determinata dal segno dell'elemento $(\bar{i_s}, \bar{j_s})$ della matrice $\boldsymbol{W}$. Se questo termine è minore di 0, allora POS=0, altrimenti POS=1;
- L_out, insieme uguale all'insieme in ingresso L se l'elemento $(\bar{i_s}, \bar{j_s})$ della matrice $\boldsymbol{W}$ è maggiore di 0, altrimenti, se questo termine è minore di 0, l'insieme L_out è uguale all'insieme L in ingresso a cui è stato tolto l'elemento $\bar{j_s}$.
- U_out, insieme uguale all'insieme in ingresso U se l'elemento $(\bar{i_s}, \bar{j_s})$ della matrice $\boldsymbol{W}$ è minore di 0, altrimenti, se questo termine è maggiore di 0, l'insieme U_out è uguale all'insieme U in ingresso a cui è stato tolto l'elemento $\bar{j_s}$.

La funzione *calcola_LU* assegna a ogni step alla matrice *LU_out* un solo elemento, sia esso un elemento della prima riga (un lower bound) o un elemento della seconda riga (un upper bound), valutando il segno dell'elemento $\boldsymbol{W}(\bar{i_s}, \bar{j_s})$. Il valore del bound è cosi determinato:

1. se $\boldsymbol{W}(\bar{i_s}, \bar{j_s}) < 0$ allora si assegna il valore $-\tau_s$ al termine della matrice $\mathbf{LU}$ nella prima riga e nella colonna $\bar{j_s}$; quindi si avrà NEG=1, POS=0 e l'elemento $\bar{j_s}$ verrà sottratto dall'insieme L;
2. se $\boldsymbol{W}(\bar{i_s}, \bar{j_s}) > 0$ allora si assegna il valore τ_s al termine della matrice $\mathbf{LU}$ nella seconda riga e nella colonna $\bar{j_s}$; quindi si avrà NEG=0, POS=1 e l'elemento $\bar{j_s}$ verrà sottratto dall'insieme U.

Prima di concludere si vuole precisare che per poter eliminare un elemento da un insieme L o U, è stato necessario creare un'altra funzione, o meglio, due funzioni con uguale codice: una rivolta all'aggiornamento dell'insieme L chiamata *trova_L* e una per aggiornare l'insieme U, chiamata *trova_U*. Queste funzioni sono state richiamate all'interno della funzione *calcola_LU* per determinare l'insieme L e l'insieme U quando da uno di questi due insiemi deve essere eliminato l'elemento $\bar{j_s}$.

Si noti che al termine dell'applicazione dell'algoritmo i due insiemi non conterranno alcun elemento poichè a tutti i posti è stato assegnato sia un lower bound che un upper bound.

Esempio 4.3.4. *Sia* $\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$, $\tau_s = 1$, $\mathbf{IND} = \begin{bmatrix} 2 & 1 \end{bmatrix}$, *si supponga sia* $\mathbf{LU} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ *una matrice di elementi nulli, sia* $L = \begin{bmatrix} 1 & 2 \end{bmatrix}$ *e* $U = \begin{bmatrix} 1 & 2 \end{bmatrix}$.
Si determini la matrice LU_out in uscita.

La funzione usata per calcolare tale matrice è calcola_LU.

Si pone $\bar{i_s} = \mathbf{IND}(\mathbf{1}, \mathbf{1}) = \mathbf{2}$ *e* $\bar{j_s} = \mathbf{IND}(\mathbf{1}, \mathbf{2}) = \mathbf{1}$*; si valuta il termine* $\boldsymbol{W}(\bar{i_s}, \bar{j_s}) = \boldsymbol{W}(2, 1)$ *il quale risulta uguale a* -2 *dunque ci si trova nel caso* $\boldsymbol{W}(\bar{i_s}, \bar{j_s}) < 0$*; quindi si ha NEG=1, POS=0 e si determina un elemento* $l_s(p_{\bar{j_s}}) = -\tau_s$ *da posizionare nella prima riga della matrice LU_out (poichè si tratta di un lower bound) e nella colonna* $\bar{j_s} = 1$*; risulta LU_out* $= \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$.

4.3.5 Determinazione degli insiemi L e U

Ciò che verrà detto per la funzione *trova_L* vale per la funzione *trova_U*. Queste funzioni sono identiche, ciò che cambia è solo l'insieme in ingresso a cui si fa riferimento.

La loro sintassi è:

[L_out]=trova_L(L,IND).

Gli argomenti in ingresso sono:

- L, vettore $1 \times m$ in cui ciascun elemento indica un posto;
- **IND**, prima riga della matrice $\boldsymbol{J}$ che contiene l'indice di riga e di colonna di un elemento minimo della matrice $\boldsymbol{S}$.

Si associa a $\bar{j_s}$ l'indice di colonna, cioè l'elemento (1,2) della matrice $\boldsymbol{J}$.

L'argomento in uscita è:

- L_out, il nuovo insieme privato dell'elemento $\bar{j}_s$.

La funzione *find* ricerca, all'interno dell'insieme L, l'elemento corrispondente al posto $\bar{j}_s$ e restituisce l'indice di riga chiamato *index*.

Si possono verificare le seguenti situazioni.

Se l'insieme L ha un solo elemento ($m = 1$), eliminando quell'unico elemento $\bar{j}_s$ dall'insieme, L_out risulta uguale a 0.

Se l'insieme L ha più di un elemento e se $index = 1$, allora il nuovo insieme ottenuto in uscita sarà uguale a tutto ciò che va dall'elemento di indice 2 all'elemento di indice m.

Se L ha più di un elemento e se $index = m$, allora l'insieme in uscita sarà uguale all'insieme in ingresso fino all'elemento di indice $m - 1$.

Se l'insieme L ha più di un elemento e se $index$ non si trova in nessuna delle condizioni descritte sopra, allora l'insieme L_out in uscita è così determinato:

1. per gli elementi di indici compresi tra 1 e $index - 1$, L_out è uguale all'insieme in ingresso L;
2. per gli elementi di indici compresi tra $index + 1$ e m, L_out è uguale all'insieme in ingresso ma traslato di una posizione in avanti.

4.3.6 Calcolo di aggiornamento delle matrici

Quest'ultima funzione permette di determinare la matrice $\boldsymbol{W}$ e il vettore $\boldsymbol{k}$ che andranno in ingresso allo step successivo dell'algoritmo.

La sintassi della funzione è la seguente:

[W_new,k_new]=calcola_W_k(W,k,tau_s,IND,NEG,POS).

Come si può vedere nessuno degli argomenti in ingresso è nuovo, ma sono tutti definiti nello script *algoritmo_HB* o per mezzo di altre funzioni.

Gli argomenti in uscita sono:

- *W_new*, matrice $n_c \times m$ i cui elementi sono aggiornati in seguito alla determinazione del lower bound o dell'upper bound;
- *k_new*, vettore colonna con le stesse dimensioni del vettore $\boldsymbol{k}$ in ingresso.

Questa funzione, noto il valore dell'indice $\bar{j}_s$, valuta l'elemento $\boldsymbol{W}(i, \bar{j}_s)$ e distingue i seguenti casi:

1. CASO 1: se $\boldsymbol{W}(i, \bar{j}_s) < 0$ e NEG=1, allora k_i e w_i vengono determinati come espresso allo step 2.5 dell'Algoritmo HB, facendo attenzione al fatto che, se l'indice j fosse uguale a $\bar{j}_s$, allora la matrice *W_new* avrebbe un elemento nullo nella riga i-esima e nella colonna j-esima;
2. CASO 2: se $\boldsymbol{W}(i, \bar{j}_s) > 0$ e POS=1, allora k_i e w_i vengono determinati come espresso allo step 2.5 dell'Algoritmo HB, facendo attenzione al fatto che, se l'indice j fosse uguale a $\bar{j}_s$, allora la matrice *W_new* avrebbe un elemento nullo nella riga i-esima e nella colonna j-esima;
3. CASO 3: se invece non ci si ritrova in nessuno dei due casi precedenti, allora gli elementi k_i e w_i sono uguali a quelli presenti in ingresso in quella stessa posizione.

Esempio 4.3.5. *Sia* $\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$, $\boldsymbol{k} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$, $\mathbf{IND} = \begin{bmatrix} 2 & 1 \end{bmatrix}$,
NEG=1, POS=0.
Determinare la matrice W_new *e il vettore* k_new.

In questo caso si usa la funzione calcola_W_k. *Noto* $\bar{j}_s = 1$, *si valuta l'elemento di ogni riga che si trova nella colonna* $\bar{j}_s = 1$ *della matrice* $\boldsymbol{W}$ *e, sapendo che NEG=1 e POS=0, si decide a quale caso appartenere. Per*

1. $i = 1$, $\boldsymbol{W}(i, \bar{j}_s) = \boldsymbol{W}(1, 1) = 1$, *NEG=1, POS=0, si applica il CASO 3;*
2. $i = 2$, $\boldsymbol{W}(i, \bar{j}_s) = \boldsymbol{W}(2, 1) = -2$, *NEG=1, POS=0, si applica il CASO 1;*
3. $i = 3$, $\boldsymbol{W}(i, \bar{j}_s) = \boldsymbol{W}(3, 1) = -1$, *NEG=1, POS=0, si applica il CASO 1;*
4. $i = 4$, $\boldsymbol{W}(i, \bar{j}_s) = \boldsymbol{W}(4, 1) = 0$, *NEG=1, POS=0, si applica il CASO 3.*

Una volta deciso che caso applicare si ottiene $\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}$, $\boldsymbol{k} = \begin{bmatrix} 4 \\ 1 \\ 1 \\ 1 \end{bmatrix}$.
Questi costituiscono i dati in ingresso allo step successivo dell'algoritmo.

4.3.7 Un nuovo approccio

Prima di concludere il discorso sull'implementazione dell'Algoritmo HB si vogliono fare delle precisazioni relative ad alcuni cambiamenti apportati rispetto a quanto proposto in [1].

Il cambiamento introdotto in questa tesi, porta a una riduzione del numero di funzioni da richiamare nello script *algoritmo_HB*. Risulta non rilevante la presenza delle funzioni *trova_L* e *trova_U* che permettono di definire i nuovi insiemi di elementi candidati a ricevere l'attribuzione del bound. L'unione degli elementi di questi insiemi faceva variare l'indice di colonna della matrice $\boldsymbol{S}$ trovata per mezzo della funzione *calcola_S* allo step 2.3 dell'algoritmo.

Se gli elementi trovati con l'espressione (4.2), come supposto, costituiscono una matrice, nel caso in cui nell'insieme $L \cup U$ dovesse mancare l'indice relativo a una colonna (quindi quell'unione di elementi è gia stato privato di 2 o più elementi) allora la matrice $\boldsymbol{S}$ non sarebbe più una matrice $n_c \times m$, non si avrebbe più la dimensione desiderata e quindi si perderebbe informazione su una possibile coppia di indici di un valore minimo.

Perciò nel determinare la matrice $\boldsymbol{S}$ con la funzione *calcola_S* si è eliminata la condizione $j \in L_s \cup U_s$, sostituendola con la più semplice condizione $j \in 1, .., m$ che permette di avere a ogni step una matrice $n_c \times m$. Questo cambiamento può far pensare che questa nuova soluzione possa più volte attribuire a uno stesso posto un bound, poichè si può scegliere una stessa coppia di indici $(\bar{i_s}, \bar{j_s})$ più volte. Si è verificato che questa condizione non può verificarsi mai: infatti $(\bar{i_s}, \bar{j_s})$ sono indici di un valore minimo presente all'interno della matrice $\boldsymbol{S}$. Dall'espressione (4.2) si nota che ogni elemento ha al denominatore il valore assoluto dell'elemento della matrice $\boldsymbol{W}$ alla riga i e alla colonna j. Tale elemento è sempre uguale a 0 per la riga $i = i_s$ e la colonna $j = j_s$. Quindi la divisione per 0 non potrà mai condurre a un valore minimo della matrice $\boldsymbol{S}$ e quindi in quella stessa posizione allo step successivo non si troverà mai un valore minimo.

4.4 Implementazione dell'Algoritmo HMB

Questo algoritmo non presenta profonde differenze di calcolo rispetto all'Algoritmo HB, ma questi due metodi risolutivi non possono considerarsi sostitutivi l'uno dell'altro: infatti bisogna precisare che non si può decidere di applicare l'uno o l'altro indistintamente, poichè per applicare l'Algoritmo HMB bisogna conoscere i dati in uscita dall'Algoritmo HB, che costituiscono il punto di partenza per le modifiche da apportare al fine di ricercare il box intero massimale. Inoltre vi è un importante relazione che deve essere tenuta in considerazione e che deve verificarsi per poter procedere ad applicare il secondo algoritmo. Tale condizione, come già discussso, è rappresentata dall'equazione (3.17), la quale afferma che, se la sequenza è strettamente crescente, con la possibilità di avere valori uguali nella coda della sequenza, allora si può ottenere un box intero massimale.

Riassumendo, per poter eseguire l'Algoritmo HMB bisogna prima avere eseguito l'Algoritmo HB e verificato che la sequenza di τ sia strettamente crescente assumendo la forma indicata in (3.17).

4.4.1 Un nuovo script per l'Algoritmo HB

Si può pensare di prendere il file chiamato *algoritmo_HB* dell'omonimo algoritmo implementato e di aggiungere a questo delle funzioni che permettano di calcolare il valore dei nuovi termini che costituiranno i boundaries. Innanzitutto bisogna apportare delle modifiche che permettano di conoscere la sequenza di valori di τ e poi di tenere memoria della sequenza di indici $\bar{j}_s$ con cui vengono assegnati i boundaries.

A tale scopo lo script *algoritmo_HB* viene modificato introducendo due nuove variabili rappresentate da due vettori riga nulli di dimensione $1 \times (2m)$, di identiche dimensioni in cui verranno, a ogni step, memorizzati i valori del parametro τ_s e dell'indice $\bar{j}_s$. *vettore_tau* memorizza il valore di τ trovato a ogni step. *vettore_js* memorizza gli indici $\bar{j}_s$ scelti a ogni step. Entrambe le variabili vengono memorizzate prima di concludere l'esecuzione dello step s dell'algoritmo.

La conoscenza del vettore *vettore_tau* è molto importante per due aspetti:

1. perchè permette di individuare l'indice μ della sequenza di τ in cui tale sequenza smette di essere strettamente crescente e diventa costante;

2. noto μ sarà possibile fissare il numero di step dell'Algoritmo HMB.

4.4.2 Algoritmo HMB

Per la ricerca del box intero massimale si crea un nuovo script chiamato *algoritmo_HMB* in cui si richiama il file *algoritmo_HB* modificato che ha permesso di determinare lower bound e upper bound e quindi i risultati di tale file, cioè la matrice dei boundaries $\mathbf{LU}$. Inoltre vengono richiamate due funzioni: *trova_mu* e *trova_X* che determinano rispettivamente l'indice μ che dà informazioni sul numero di step dell'Algoritmo HMB e la matrice $\boldsymbol{X}$, usata per determinare la modifica da apportare ai termini coinvolti.

Per trovare l'indice μ si è creata una funzione *trova_mu* la cui sintassi è:

[mu]=trova_mu(vettore_tau).

L'argomento in ingresso è:

- il vettore riga *vettore_tau* che indica i valori di τ assunti ai differenti step durante l'esecuzione dell'Algoritmo HB.

In uscita si ottiene:

- l'indice μ che, diminuito di un unità, indica su quanti elementi verrà fatta un'ulteriore analisi per rendere l'ipercubo massimale.

Tale funzione ricerca all'interno del vettore *vettore_tau* il valore massimo, chiamato MASSIMO_TAU, con l'ausilio della funzione base *find* e, sempre con questa funzione si cercano gli indici in cui compare la variabile MASSIMO_TAU nel vettore *vettore_tau*. Poichè è possibile che tale valore massimo sia presente più di una volta nella coda della sequenza, si indica con μ il primo indice trovato.

Infine nello script creato per questo algoritmo vi sarà anche una nuova funzione chiamata *trova_X* che permette di calcolare quanto definito in (3.8).

Tale funzione ha seguente sintassi:

[X]=trova_X(A,LU).

Gli argomenti in ingresso sono:

- $\boldsymbol{A}$, matrice $n_c \times m$ data in ingresso allo script *algoritmo_HB*;
- $\mathbf{LU}$, matrice dei boundaries trovati con l'Algoritmo HB.

Ciò che si vuole determinare in uscita è:

- $\boldsymbol{X}$, matrice $n_c \times m$ definita come in (3.8).

Questa funzione valuta il segno di ogni elemento della matrice $\boldsymbol{A}$ e, se:

- l'elemento $\boldsymbol{A}(i,j) > 0$, assegna all'elemento (i,j) della matrice $\boldsymbol{X}$ il valore che l'upper bound (seconda riga della matrice $\mathbf{LU}$) ha nella colonna j;
- l'elemento $\boldsymbol{A}(i,j) \leq 0$, assegna all'elemento (i,j) della matrice $\boldsymbol{X}$ il valore che il lower bound (prima riga della matrice $\mathbf{LU}$) ha nella colonna j.

Riassumendo, nello script per l'Algoritmo HMB, denominato *algoritmo_HMB* sono richiamate le seguenti funzioni:

- lo script *algoritmo_HB* dell'Algoritmo HB con l'aggiunta delle due nuove variabili *vettore_js* e *vettore_tau*;
- la funzione *trova_mu*;
- la funzione *trova_X*.

Dopo aver richiamato queste funzioni si impone l'esecuzione di un ciclo che si ripete $\mu - 1$ volte in cui viene definito il valore di $\bar{j}_s$. Questo valore è definito scegliendo l'elemento del vettore *vettore_js* che si trova nella colonna indicata dallo step che si sta eseguendo.

Oltre a tale assegnamento, durante ogni ciclo verranno richiamate ed eseguite le seguenti funzioni:

- *trova_INT_MIN* che determina di quanto il lower bound deve essere diminuito o l'upper bound deve essere aumentato;
- *trova_POS* decide che valore assume la variabile POS;
- *trova_LU_max* che determina i boundaries dell'ipercubo massimale.

4.4.3 Variabile POS

L'algoritmo trattato viene eseguito in un numero di step pari a $\mu - 1$ dove, come detto nel paragrafo precedente l'indice μ viene determinato per mezzo della funzione *trova_mu*.

La costruzione dei boundaries che portano a un box intero massimale considera il primo elemento del vettore *vettore_js* e esegue un confronto tra il valore di tale elemento e l'indice generico j che varia da 1 a m e indica uno tra i possibili posti della GMEC.

1. Se risulta $j \neq \bar{j_s}$, allora i valori di lower bound e upper bound restano invariati.
2. Se invece si verifica $j = \bar{j_s}$, allora si deve valutare l'upper bound relativo al posto indicato da $\bar{j_s}$.

Il secondo caso rappresenta il punto di partenza per la determinazione dei nuovi boundaries, poichè determina la variabile booleana POS, la quale, a seconda del suo valore, aggiorna il valore del lower bound (POS=0) o quello dell'upper bound (POS=1).

Questi passi vengono eseguiti all'interno della funzione *trova_POS*, la cui sintassi è:

[POS]=trova_POS(LU,js,vettore_tau_s).

Gli argomenti in ingresso e in uscita sono già stati tutti definiti e trattati per cui non ci si sofferma sulle loro caratteristiche o sul ruolo che essi rivestono, ma su ciò che essi compiono all'interno della funzione e come vengono determinati.

La funzione *trova_POS* determina il valore della variabile POS utilizzando l'elemento $\bar{j_s}$ della seconda riga della matrice $\mathbf{LU}$. Se tale termine, ovvero $u(p_{\bar{j_s}})$ risulta uguale al valore di τ_s, cioè all'elemento del vettore *vettore_tau* nella colonna indicata dallo step che si sta eseguendo, allora POS=1, altrimenti POS=0.

Si noti che, a differenza di quanto proposto in [1] in cui si usa la variabile POS=1 se $u(p_{\bar{j_s}}) = \tau_s$ o NEG=1 se ciò non si verifica, si è scelto di usare una sola variabile (POS) che può assumere valore 1 se l'uguaglianza è verificata oppure 0 altrimenti.

4.4.4 Fattore di incremento o decremento

E' necessario conoscere di quanto un upper bound possa essere incrementato o un lower bound possa essere decrementato al fine di trovare i boundaries che rendono massimale il box interno determinato con l'Algoritmo HB.

Per conoscere questa quantità è stata creata la funzione *trova_INT_MIN*, la cui sintassi è:

[INT_MIN]=trova_INT_MIN(W,k,js,X).

Poichè degli argomenti in ingresso e in uscita si è gia discusso e si conoscono le proprietà, si vuole solo precisare che la matrice $\boldsymbol{W}$ considerata, a differenza di quanto si potrebbe pensare di ottenere in uscita dell'Algoritmo HB, non è la matrice *W_new* ma è proprio la matrice $\boldsymbol{W}$ determinata con la funzione *data_input*. La stessa considerazione può essere estesa anche al vettore $\boldsymbol{k}$.

Il termine in uscita INT_MIN viene calcolato applicando la seguente formula:

$$\left\lfloor \frac{k_i - \tilde{\boldsymbol{w}}_i \cdot \bar{\boldsymbol{x}}_i^T}{|\tilde{w}_i(p_{\bar{j}_s})|} \right\rfloor . \tag{4.3}$$

Lo scalare INT_MIN è il minimo valore tra tutti gli interi trovati applicando l'operatore *floor* a un rapporto. Al numeratore di questo rapporto vi è una differenza tra l'elemento k_i e lo scalare ottenuto da un prodotto vettoriale tra il vettore riga $\boldsymbol{w}_i$ e la riga $i-$esima della matrice $\boldsymbol{X}^T$; al denominatore vi è il valore assoluto dell'elemento $(i, \bar{j}_s)$ della matrice $\boldsymbol{W}$. Per cui la funzione *trova_INT_MIN*, dopo aver determinato il valore assoluto dell'elemento $(i, \bar{j}_s)$ della matrice $\boldsymbol{W}$ in ingresso, esegue delle operazioni di sottrazione, moltiplicazione e divisione tra i diversi termini espressi nell'equazione (4.3). Per ogni indice i si ottiene un valore intero di tale rapporto, tra tutti questi INT_MIN è uguale a quello di valore minimo.

4.4.5 Determinazione dei nuovi boundaries

Nota la variabile POS e il fattore di incremento o decremento INT_MIN, si sa se deve essere aggiornata la prima riga o la seconda riga della matrice $\mathbf{LU}$ in corrispondenza della colonna indicata da $\bar{j}_s$ e quale sarà l'effetto di tale aggiornamento. Il valore assunto dal bound sarà trovato con la funzione *trova_LU_max*.

Questa funzione ha la seguente sintassi:

[LU_max]=trova_LU_max(W,LU,INT_MIN,js,vettore_tau,POS,s).

Molti degli argomenti in ingresso sono già stati discussi, come $\boldsymbol{W}$, $\mathbf{LU}$, INT_MIN, j_s, *vettore_tau* e POS. L'unico elemento in ingresso non ancora incontrato è:

- s, il quale rappresenta lo step d'esecuzione ed è indicato esplicitamente perchè servirà a determinare quale elemento del vettore *vettore_tau* si dovrà considerare.

In uscita si avrà:

- la matrice *LU_max* con le stesse dimensioni della matrice $\mathbf{LU}$ in ingresso; la prima riga di questa matrice è la riga dei lower bounds e viene indicata con $\boldsymbol{l}^*$, la seconda riga è quella degli upper bounds e si indica con $\boldsymbol{u}^*$.

La funzione *trova_LU_max* associa un valore a ogni elemento della matrice $\mathbf{LU}$, distinguendo tra:

valori di j diversi da $\bar{j}_s$: in questo caso gli elementi della matrice $\mathbf{LU}$ non vengono modificati;

valori di j uguali a $\bar{j}_s$: questa condizione porta a due casi differenti che assegnano un upper bound al posto $\bar{j}_s$ se POS=1, altrimenti a quello stesso posto viene assegnato un lower bound.

I nuovi lower bound e upper bound sono determinati semplicemente con una somma o una sottrazione, ma possono anche in questo caso non essere modificati. Per sapere se il lower bound o l'upper bound possano essere rispettivamente diminuito o aumentato bisogna eseguire un confronto tra il valore di τ allo step s e il valore di τ allo step successivo.

La funzione *trova_LU_max*, noto l'indice $\bar{j}_s$ e la variabile POS, si limita a indirizzare verso l'incremento dell'upper bound (POS=1) se gli elementi di *vettore_tau* risultano costanti, altrimenti, se questi risultano crescenti, quel bound non subisce nessun'evoluzione.

Lo stesso discorso può essere ripetuto per il lower bound (POS=0), in cui, se gli elementi di *vettore_tau* risultano costanti, allora quel bound è decrementato rispetto al valore precedente di una quantità indicata da INT_MIN, altrimenti resta uguale.

4.5 Implementazione del caso non singleton

I risultati dell'Algoritmo HMB possono essere usati per determinare un insieme massimale di marcature legali decentralizzate, come è stato dimostrato nell'Esempio (3.5.1). Perciò dalla conoscenza dei vettori $\boldsymbol{l}^*$ e $\boldsymbol{u}^*$ e nota la ripartizione di posti, è possibile definire degli insiemi di marcature legali decentralizzate come esposto in (3.18). Ciò significa che partendo dai risultati dell'Algoritmo HMB si è in grado di trovare un insieme massimale di marcature decentralizzate, aggiungendo al file *algoritmo_HMB* dell'algoritmo che ottiene il box massimale, una funzione che esegua quanto espresso in (3.18).

Si crea uno script in cui si richiamano i risultati precedenti: *algoritmo_HB* e *algoritmo_HMB*. I dati forniti in ingresso a questo script, chiamato *non_singleton*, sono la matrice *W_in* determinata dalla GMEC globale e la matrice $\boldsymbol{P}$ che rappresenta la suddivisione dei posti supponendo che siano non singleton. Gli elementi della matrice $\boldsymbol{P}$ possono assumere solo i valori pari a 1 oppure 0; la matrice $\boldsymbol{P}$ avrà 1 nella riga i e nella colonna j se il posto p_j appartiene all'insieme P_i, 0 altrimenti.

La funzione creata per determinare la matrice *W_out* e il vettore *k_out* che definiranno le GMEC decentralizzate, ha la seguente sintassi:

[W_out,k_out]=trova_W_out_kout(W,P,X).

Gli argomenti in ingresso sono:

- *W_in*, matrice definita dalla GMEC globale. Ha tante righe quante sono le GMEC singole e tante colonne quanti sono i posti;
- $\boldsymbol{P}$, matrice con un numero di righe uguale al numero di insiemi non singleton P_i e un numero di colonne uguale al numero di posti indicati dalle GMEC;
- $\boldsymbol{X}$, matrice $n_c \times m$ definita come in (3.8) a seconda dei valori dei boundaries.

Gli argomenti in uscita sono:

- la matrice *W_out* e il vettore *k_out* che determinano un nuovo insieme di GMEC decentralizzate.

La matrice *W_in* è costituita da q righe; ogni riga di *W_in* in uscita sarà convertita in ν righe, dove ν è il numero di righe della matrice $\boldsymbol{P}$. Perciò la matrice *W_out* avrà un numero di colonne pari al numero di colonne di *W_in* e di $\boldsymbol{P}$ e un numero di righe pari a $\nu \times q$. Ogni riga della matrice *W_in* dovrà 'scansionare' l'intera matrice $\boldsymbol{P}$ per sapere se il posto p_j appartiene all'insieme P_i.

Quindi la funzione valuta l'elemento (i, j) della matrice $\boldsymbol{P}$: se questo elemento risulta uguale a 1, cioè il posto p_j appartiene all'insieme P_i, allora il termine $(i + (r - 1) * nu, j)$ della matrice *W_out* è uguale al termine (r, j) della matrice *W_in* in ingresso, altrimenti, se la condizione di appartenenza non è verificata, l'elemento $(i + (r - 1) * nu, j)$ della matrice *W_out* vale 0.

Gli elementi del vettore *k_out* vengono calcolati solo dopo aver determinato la matrice *W_out*, poichè si richiede il prodotto tra la riga $(i + (r - 1) * nu)$ della matrice *W_out* per la trasposta della r-esima riga della matrice *X_new*, la quale contiene gli stessi elementi della matrice $\boldsymbol{X}$ ma assume le dimensioni della matrice *W_in*. Con tale moltiplicazione si ottiene un vettore *k_out* riga che sarà trasposto per ottenere la GMEC desiderata.

4.5.1 Bound traslati

Un'ultima funzione è stata creata al fine di ottenere le coordinate dei boundaries nel sistema di riferimento originale. Infatti si ricorda che per eseguire gli algoritmi descritti, si è suppposto di considerare una differente coppia $(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$ rispetto a quella definita dalla GMEC iniziale $(\boldsymbol{W}, \boldsymbol{k})$. A partire dalla conoscenza del termine *X_opt* che rappresenta il punto in cui viene traslata l'origine del sistema, si procede a determinare la nuova coppia, il cui procedimento è già stato discusso.

Con la funzione *trasla_LU* si vogliono trovare le coordinate dei boundaries nell'originale sistema il cui centro è il punto di coordinata 0. Questa funzione ha la seguente sintassi:

[LU_def]=trasla_LU(LU,X_opt).

Gli argomenti in ingresso, $\mathbf{LU}$ e *X_opt* sono termini calcolati con funzioni precedentemente descritte e discusse.

L'unico argomento in uscita è:

- la matrice *LU_def* che presenta le stesse caratteristiche della matrice in ingresso, ma che come già detto farà riferimento a un diverso sistema. Quindi, come suggerito dall'abbreviazione def *(definitivo)*, questa sarà la matrice che dovrà essere considerata quando si fa riferimento alle coordinate dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Questa funzione provvede a eseguire una traslazione del tipo $m = m^{'} + c$ sulla base della conoscenza del vettore *X_opt* che rappresenta il centro $\boldsymbol{c}$ del vecchio sistema e della matrice $\mathbf{LU}$ che rappresenta le coordinate dei boundaries di un ipercubo intero interno a $\mathcal{M}(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{k}})$.

4.6 Conteggio marcature

Un aspetto particolarmente interessante è la possibilità di conoscere quante marcature legali $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{W} \cdot \boldsymbol{m} \leq \boldsymbol{k}\}$ sono comprese all'interno dell'ipercubo intero massimale. Ciò significa che all'interno dello spazio consentito, definito dai boundaries determinati con la funzione *trasla_LU* si devono contare quanti vettori colonna $\boldsymbol{M}$ costituiti da un numero di righe pari al numero di posti definiti nelle GMEC contengono solo elementi positivi o nulli. La presenza di un elemento negativo renderebbe la marcatura $\boldsymbol{M}$ non raggiungibile, poichè inesistente dal momento che una marcatura deve essere definita da un numero intero non negativo.

Riuscire a contare le marcature presenti internamente all'ipercubo massimale risulta semplice solo se il numero di posti è al più uguale a due; ma avendo anche solo tre posti l'enumerazione diventa piuttosto complicata.

Ogni posto, come precedentemente specificato, affinchè gli algoritmi siano applicabili, deve essere vincolato almeno una volta, cioè ogni posto deve comparire almeno in una GMEC. Questo aspetto è di grosso aiuto perchè limita superiormente i valori assunti da ciascun posto, in modo che lo spazio ammissibile definito dall'insieme di GMEC sia sempre limitato inferiormente e superiormente e quindi anche il conteggio delle marcature può scansionare i valori che quel posto può assumere senza dover mai arrivare all'infinito.

Per eseguire tale conteggio si impone la ricerca di marcature legali comprese tra il valore minimo e il valore massimo che delimitano ciascun posto; si deve verificare per tutti i valori interi compresi tra il lower bound e l'upper bound, per tutti i posti contemporaneamente, quante marcature legali sono comprese in quello spazio.

Si supponga di avere un sistema con due soli posti. Fissato un valore intero per il posto $m(p_1)$ corrispondente al lower bound, scelto anche per il posto $m(p_2)$ il limite inferiore dell'ipercubo, si verifica se quella coordinata, candidata a diventare una marcatura, sia interna allo spazio ammissibile. Se ciò si verifica, cioè se risulta che $\boldsymbol{W} \cdot \boldsymbol{m} \leq \boldsymbol{k}$, il conteggio avanza di uno. Successivamente si considera sempre lo stesso valore per il posto $m(p_1)$ e avanza la coordinata relativa al posto $m(p_2)$ e si ripete lo stesso ragionamento, fino ad esaurire tutte le combinazioni di valori interi presenti nell'ipercubo.

Il codice scritto a tale scopo non è generalizzabile per un numero di posti m non noto, per cui si ha l'esigenza di creare sempre un nuovo script, chiamato *conta_marc_numeroposti* per ogni esercizio diverso, o meglio, per ogni esercizio in cui cambia il numero di posti. Infatti ogni script si riferisce a un sistema con un fissato numero di posti.

Questa tecnica può essere estesa a un sistema con un numero di posti qualsiasi.

4.6.1 File d'esecuzione

Nello script vengono richiamati tutti i risultati ottenuti precedentemente: i boundaries dell'ipercubo massimale e gli insiemi di marcature legali decentralizzate, oltre che la matrice *W_in* e il vettore *k_in* che definiscono la GMEC. Oltre questi dati si sottolinea l'introduzione di una variabile chiamata *marcature* che segna il conteggio eseguito dal programma, ed è inizialmente posta uguale a 0 e ogni volta che trova una marcatura all'interno dell'ipercubo, avanza di uno; inoltre vi è un vettore riga $\mathbf{Valid}$ composto da un numero di colonne uguale al numero righe della matrice *W_in*, i cui elementi sono inizialmente tutti nulli.

Dopo aver introdotto i risultati precedenti si è scelto di rinominare la matrice *LU_def* con il nome che, per semplicità sembra essere più adeguato in questo file, cioè $\mathbf{LU}$.

La matrice $\mathbf{LU}$ va in ingresso alla funzione *limita_bound* la cui sintassi è:

[LU]=limita_bound(LU).

L'unico termine in ingresso è:

- la matrice $\mathbf{LU}$ che, come detto, non è altro che la matrice *LU_def* determinata con la funzione *trasla_LU*.

In uscita si ottiene:

- la matrice $\mathbf{LU}$ dove però compaiono solo elementi positivi o nulli.

Infatti la funzione è implementata in modo da valutare ciascun elemento (i, j) della matrice $\mathbf{LU}$; se l'elemento (i, j) assume valore negativo, allora sarà sostituito da un elemento nullo, se assume valore positivo o nullo rimane uguale. Il fine di questa funzione è quello di non trovare un ipercubo che abbia coordinate negative, e che quindi possa racchiudere solo marcature con valori non negativi.

Per scrivere il programma che conta le marcature presenti all'interno dell'ipercubo, sono state usate tante variabili, chiamate genericamente $x_1, x_2, x_3, \ldots x_m$, quante sono i posti appartenenti all'insieme di GMEC. A partire dalle variabili di indice più basso, e procedendo verso quelle di indice più alto, si definisce per ognuna l'intervallo in cui la variabile può variare, cioè il valore intero minimo e il valore intero massimo che essa può assumere. Le prime a variare sono quelle di indice più alto e, solo dopo che queste hanno assunto tutti i possibili valori che le definiscono, può iniziare a variare la variabile con indice immediatamente inferiore. Si procede in questo modo fino a quando tutte le m variabili hanno assunto tutti i valori.

Il valore assunto da ciascuna variabile viene memorizzato in un vettore colonna $\boldsymbol{X}$, il quale viene poi moltiplicato per la matrice *W_in*, e si verifica la condizione $\textit{W\_in} \cdot \boldsymbol{X} \leq \textit{k\_in}$, cioè se i valori che costituiscono il vettore $\boldsymbol{X}$ si trovano all'interno dello spazio definito dall'insieme di GMEC. Se la condizione è verificata, allora nel vettore $\mathbf{Valid}$ si inserisce un elemento unitario, altrimenti un elemento nullo. Questa verifica viene eseguita per ogni riga della matrice *W_in*, al termine di tale ciclo si esegue un'analisi del vettore $\mathbf{Valid}$. Se gli elementi di tale vettore sono tutti unitari, la variabile *marcature* avanza di uno, cioè l'insieme di coordinate che definiscono il vettore $\boldsymbol{X}$ appartiene all'ipercubo massimale ed è interno allo spazio definito dalle GMEC, quindi quell'insieme può considerarsi una marcatura raggiungibile.

Al termine della variazione di ciascuna variabile il conteggio può considerarsi concluso e viene restituito il numero di marcature presenti nell'ipercubo, cioè

quanti vettori contenenti elementi non negativi vi sono nello spazio delimitato dai boundaries.

Può essere interessante ampliare il conteggio delle marcature allo spazio definito dall'insieme di GMEC e stabilire un confronto tra il numero di marcature presenti all'interno dell'ipercubo massimale, conteggiate nel modo appena descritto e il numero di marcature presenti nello spazio delimitato dall'insieme delle GMEC. A tale scopo si può usare un programma analogo a quello usato per il conteggio delle marcature all'interno dei boundaries, ma apportando delle modifiche.

Si crea uno script, chiamato ancora *conta_marc_numeroposti* in cui non vi sarà più alcun riferimento ai boundaries e quindi a tutti i risultati precedentemente ottenuti, ma saranno definiti: la matrice *W_in* e il vettore *k_in* determinati dalle GMEC, le variabili *marcature* e **Valid** che hanno lo stesso ruolo del caso precedente, e la funzione *trova_index* la cui sintassi è:

[index]=trova_index(W_in,k_in).

Gli argomenti in ingresso sono:

- la matrice *W_in* e il vettore *k_in* determinati dalle GMEC iniziali.

L'unico argomento in uscita è:

- la matrice **index**, composta di due righe e m colonne.

Questa funzione punta a trovare, per ciascun posto, i punti in cui le GMEC intersecano i posti. Se si hanno q GMEC si hanno q punti d'intersezione tra un posto e la GMEC; tra tutti questi punti d'intersezione, è utile conoscere solo il valore che delimita il posto inferiormente e il valore che delimita il posto superiormente. Quindi la funzione *trova_index* seleziona solo il valore minimo che viene posizionato nella prima riga e il valore massimo che viene sistemato nella seconda riga. La matrice **index** contiene nella prima riga tutti gli elementi minimi determinati durante l'esecuzione della funzione, dal posto 1 al posto m, nella seconda riga contiene gli elementi massimi, dal posto 1 al posto m.

Tutti i valori che costituiscono la matrice **index** sono valori interi, poichè determinati dal rapporto tra l'elemento i-esimo del vettore *k_in* e l'elemento (i, j) della matrice *W_in*, a cui viene applicato l'operatore floor, il quale determina il

valore intero di tale rapporto. Tutti questi valori interi calcolati sono poi selezionati secondo il criterio descritto sopra, che prende il valore minimo e massimo per ciascun posto.

Partendo dalla conoscenza della matrice **index** è possibile procedere al conteggio, poichè si hanno tutte le informazioni necessarie che consentono di delimitare lo spazio all'interno del quale si vuole eseguire il calcolo delle marcature.

Il programma non si differenzia tanto dal precedente, facente riferimento ai boundaries. Infatti l'unica differenza consiste nel campo di variazione di ciascuna variabile $x_1, x_2, \ldots, x_m$ associata ai diversi posti. Mentre nel programma precedente si fa variare ogni variabile relativa a un fissato posto, tra il limite inferiore e il limite superiore dell'ipercubo massimale rispetto a quello stesso posto, in questo caso si fa variare ogni variabile tra il centro del sistema, cioè il punto 0 e il valore massimo in cui una GMEC interseca il posto.

Capitolo 5

Simulazioni

L'attenzione di questo capitolo è rivolta a testare le funzionalità del programma scritto su esempi che presentano differenti caratteristiche. In particolare verranno considerati cinque casi: i primi due sono esempi didattici e non presentano alcun significato fisico, almeno per il modo in cui sono stati considerati in questo lavoro; il terzo e il quarto esempio invece sono un'applicazione a un sistema reale descritto da un processo di workflow; il quinto e ultimo esempio infine prende in considerazione una rete con numero di posti parametrico al fine di valutare i tempi di elaborazione.

L'applicazione di tale programma richiede la conoscenza del numero totale di posti che costituiscono la rete di Petri, senza conoscere come questi siano collegati tra loro e per mezzo di quali transizioni. Infatti nei primi due esempi considerati non si conosce la rete che essi rappresentano, ma si conosce solo un insieme di GMEC che costituiscono un insieme di vincoli sulla rete. Ciò potrebbe sembrare un vantaggio poichè non vincola alla conoscenza di un grafo, però è limitativo quando si vogliono analizzare alcune caratteristiche del sistema quale, ad esempio, il grafo di raggiungibilità della rete, che richiede la conoscenza delle connessioni tra i posti e le transizioni.

Il numero di posti che costituiscono la rete di Petri è deducibile anche senza conoscere il grafo, poichè il programma realizzato richiede che l'insieme di GMEC applicate alla rete sia tale da avere sempre almeno un vincolo su ciascun posto, ciò significa che ciascun posto sarà presente in almeno una GMEC $(\boldsymbol{w}, k)$ appartenente alla GMEC globale $(\boldsymbol{W}, \boldsymbol{k})$.

Per ciascuno dei primi quattro casi verrà risolto un problema di programmazione

lineare intero (LIPP); i dati ottenuti saranno forniti in ingresso all'Algoritmo HB e successivamente i risultati di tale algoritmo saranno in ingresso all'Algoritmo HMB, quindi si valuteranno gli insiemi massimali di marcature decentralizzate ottenute partendo dalla conoscenza dei boundaries. Infine si valuterà il numero di marcature comprese all'interno dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Eseguire tutti gli step di entrambi gli algoritmi risulta essere piuttosto lungo e elaborato rendendo complicata l'esposizione, quindi si propone di focalizzare l'attenzione su alcuni parametri interessanti per la ricerca del risultato finale e fornire di tutti gli altri solo il risultato.

5.1 Caso 1

Si consideri il seguente insieme di GMEC:

$$\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{ \quad \boldsymbol{m} \in \mathbb{N}^2 \quad | \quad m(p_1) + 2m(p_2) \leq 8, \\ -2m(p_1) + m(p_2) \leq 0\}.$$

Si determinino le coordinate dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ e il numero di marcature presenti nell'ipercubo, supponendo una suddivisione dei posti di tipo singleton.

5.1.1 Determinazione delle coordinate dell'ipercubo

Per poter applicare gli algoritmi proposti risulta di rilevante importanza la conoscenza delle coordinate del centro $\boldsymbol{c}$ e del lato 2τ dell'ipercubo intero interno massimale. Tali valori sono il risultato di un problema di programmazione lineare definito nel seguente modo:

SENSE=-1;

$$\text{C=}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix};$$

$$\text{A=}\begin{bmatrix} 1 & 2 & 3 \\ -2 & 1 & 3 \end{bmatrix};$$

B=$\begin{bmatrix} 8 \\ 0 \end{bmatrix}$;

CTYPE=$\begin{bmatrix} 'U' \\ 'U' \end{bmatrix}$;

LB=$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$;

UB=$\begin{bmatrix} Inf \\ Inf \\ Inf \end{bmatrix}$;

VARTYPE=$\begin{bmatrix} 'C' \\ 'C' \\ 'C' \end{bmatrix}$;

LPSOLVER=1.

Questo problema fornisce i seguenti risultati: *X_opt* $= \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$ e *F_opt* $= 1$.

A partire da tali informazioni e dalla conoscenza della matrice *W_in* e del vettore *k_in* che definiscono le GMEC, usando la funzione *data_input*, si determinano i dati elaborati dall'Algoritmo HB: $\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$ e $\boldsymbol{k} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$.

I valori di τ_s ai differenti step sono:

$$\tau_{s=1} = 1, \tau_{s=2} = 1, \tau_{s=3} = 1, \tau_{s=4} = 2.$$

Questi termini sono i valori assoluti che costituiscono la matrice $\mathbf{LU}$ dei boundaries che si compone ai diversi step.

Gli indici $\bar{i}_s$ e $\bar{j}_s$ scelti a ogni singolo step sono:

$$(\bar{i}_{s=1}, \bar{j}_{s=1}) = (2,2), (\bar{i}_{s=2}, \bar{j}_{s=2}) = (4,2), (\bar{i}_{s=3}, \bar{j}_{s=3}) = (2,1),$$

$$(\bar{i}_{s=4}, \bar{j}_{s=4}) = (1, 1).$$

Questi indici danno informazioni sull'ordine con cui la matrice $\mathbf{LU}$ va a comporsi. Dunque si può immediatamente dedurre che $\mathbf{LU} = \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix}$.

Infine, a ogni step si determina una nuova matrice $\boldsymbol{W}$ e un nuovo vettore $\boldsymbol{k}$ che andranno in ingresso allo step successivo. All'ultimo step i dati in uscita sono: $\boldsymbol{W} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$ e $\boldsymbol{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$.

E' possibile riassumere in maniera sintetica e con i principali dati tutti gli step eseguiti nel seguente modo:

$s = 1$

$\tau_1 = min\{\lfloor \frac{4}{3} \rfloor = \lfloor \frac{3}{3} \rfloor = \lfloor \frac{2}{1} \rfloor = \lfloor \frac{1}{1} \rfloor\} = 1,$

$J_1 = \{(1,1), (1,2), (2,1), (2,2), (4,2)\}$, si sceglie $(2,2)$

$\boldsymbol{w}_1^1 = \begin{bmatrix} 1 & 0 \end{bmatrix}$, $k_1^1 = k_1^0 - 2\tau_1 = 2$, $\quad \boldsymbol{w}_2^1 = \begin{bmatrix} -2 & 0 \end{bmatrix}$, $k_2^1 = k_2^0 - 1\tau_1 = 2,$

$\boldsymbol{w}_3^1 = \begin{bmatrix} -1 & 0 \end{bmatrix}$, $k_3^1 = k_3^0 = 2$, $\quad \boldsymbol{w}_4^1 = \begin{bmatrix} 0 & -1 \end{bmatrix}$, $k_4^1 = k_4^0 = 1,$

$\boldsymbol{u}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$, $\boldsymbol{l}_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

$s = 2$

$\tau_2 = min\{\lfloor \frac{2}{1} \rfloor = \lfloor \frac{2}{2} \rfloor = \lfloor \frac{2}{1} \rfloor = \lfloor \frac{1}{1} \rfloor\} = 1,$

$J_2 = \{(2,1), (4,2)\}$, si sceglie $(4,2)$

$\boldsymbol{w}_1^2 = \begin{bmatrix} 1 & 0 \end{bmatrix}$, $k_1^2 = k_1^1 = 2$, $\quad \boldsymbol{w}_2^2 = \begin{bmatrix} -2 & 0 \end{bmatrix}$, $k_2^2 = k_2^1 = 2,$

$\boldsymbol{w}_3^2 = \begin{bmatrix} -1 & 0 \end{bmatrix}$, $k_3^2 = k_3^1 = 2$, $\quad \boldsymbol{w}_4^2 = \begin{bmatrix} 0 & 0 \end{bmatrix}$, $k_4^2 = k_4^1 - \tau_2 = 1,$

$\boldsymbol{u}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}$, $\boldsymbol{l}_2 = \begin{bmatrix} 0 & -1 \end{bmatrix}$.

$s = 3$

$\tau_3 = min\{\lfloor\frac{2}{1}\rfloor = \lfloor\frac{2}{2}\rfloor = \lfloor\frac{2}{1}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 1,$

$J_3 = \{(2,1)\}$, si sceglie $(2,1)$

$\boldsymbol{w}_1^3 = \begin{bmatrix} 1 & 0 \end{bmatrix}, k_1^3 = k_1^2 = 2, \qquad \boldsymbol{w}_2^3 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_2^3 = k_2^2 - 2\tau_3 = 0,$

$\boldsymbol{w}_3^3 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_3^3 = k_3^2 - \tau_3 = 1, \qquad \boldsymbol{w}_4^3 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_4^3 = k_4^2 = 0,$

$\boldsymbol{u}_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \boldsymbol{l}_3 = \begin{bmatrix} -1 & -1 \end{bmatrix}.$

$s = 4$

$\tau_4 = min\{\lfloor\frac{2}{1}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{1}{0}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 2,$

$J_4 = \{(1,1)\}$, si sceglie $(1,1)$

$\boldsymbol{w}_1^4 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_1^4 = k_1^3 - \tau_4 = 0, \qquad \boldsymbol{w}_2^4 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_2^4 = k_2^3 = 0,$

$\boldsymbol{w}_3^4 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_3^4 = k_3^3 = 1, \qquad \boldsymbol{w}_4^4 = \begin{bmatrix} 0 & 0 \end{bmatrix}, k_4^4 = k_4^4 = 0,$

$\boldsymbol{u}_4 = \begin{bmatrix} 2 & 1 \end{bmatrix}, \boldsymbol{l}_4 = \begin{bmatrix} -1 & -1 \end{bmatrix}.$

Tutto ciò è il risultato dell'applicazione dell'Algoritmo HB, mentre con l'Algoritmo HMB si verifica se i dati ottenuti siano ottimali e se non lo fossero, si interviene per renderli tali.

La sequenza di lati massimali τ elaborati dall'algoritmo è $\tau_1 = 1, \tau_2 = 1, \tau_3 = 1, \tau_4 = 2$, che soddisfa la condizione (3.17), allora i vettori $\boldsymbol{l}$ e $\boldsymbol{u}$ determinano un box $\mathcal{B}(\boldsymbol{l}^*, \boldsymbol{u}^*)$ massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Si supponga di applicare anche l'Algoritmo HMB al fine di mostrare che i risultati ottenuti con l'Algoritmo HB sono effettivamente quelli che individuano il box massimale.

In questo caso si ha una sequenza di valori di $\bar{j}_s$, così come scelti precedentemente, pari a $\bar{j}_s = (2,2,1,1)$. Ciò significa che gli elementi che subiranno un incremento o un decremento sono proprio quelli che si trovano nella posizione $\bar{j}_s$. Non tutti però saranno coinvolti nella modifica, ma solo i primi 3.

Il termine di incremento o decremento è calcolato a ogni step in base alla posizione del posto candidato al cambiamento, cioè in base al valore di $\bar{j}_s$. I valori di incremento o decremento, ai diversi step, sono pari a $INT\_MIN_{s=1} = 0$, $INT\_MIN_{s=2} = 0$, $INT\_MIN_{s=3} = 0$. Questo dimostra come non vi sono cambiamenti dovuti a un aumento o una diminuzione di nessun bound, poichè la quantità di variazione è sempre nulla.

Al termine di tutti gli step la matrice $\mathbf{LU}$ risultante è $\mathbf{LU} = \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix}$.

Con la funzione *trasla_LU* si trovano i boundaries dell'ipercubo che individua l'insieme $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$. La matrice dei boundaries è $LU\_def = \begin{bmatrix} 1 & 0 \\ 4 & 2 \end{bmatrix}$.

5.1.2 Conteggio delle marcature

Il conteggio delle marcature presenti nell'ipercubo intero massimale definito da *LU_def*, le cui coordinate sono ottenute con l'Algoritmo HB e confermate dall'Algoritmo HMB, dichiara che il numero di marcature $\boldsymbol{M}$ presenti in questo spazio è pari a 12.

Anche se più elaborato, risulta interessante sapere quante marcature sono presenti nello spazio delimitato dalle GMEC. Le marcature $\boldsymbol{M}$ presenti in questo spazio sono 20.

Possiamo concludere affermando che effettivamente nei due casi sopra esposti, vi è una riduzione delle marcature; tale perdita era però attesa poichè lo spazio all'interno di cui si esegue la ricerca è diverso ed è più piccolo e ristretto nel primo caso, mentre è leggermente più grande nel secondo, anche se meno regolare dal punto di vista geometrico.

5.2 Caso 2

Si consideri il seguente insieme di GMEC:

$$\mathcal{M}(\boldsymbol{W}, \boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^3 \mid 20m(p_1) + 19m(p_2) + m(p_3) \leq 61 \\ m(p_1) + 22m(p_2) + 21m(p_3) \leq 84\}$$

Si determinino le coordinate dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$ e il numero di marcature presenti nell'ipercubo, supponendo una suddivisione dei posti di tipo singleton.

5.2.1 Determinazione delle coordinate dell'ipercubo

Il problema di programmazione lineare intero fornisce i seguenti risultati:

$X\_opt = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ e $F\_opt = 1$.

Grazie alla funzione *data_input* si trova che i dati che andranno in ingresso all'Algoritmo HB sono: $\boldsymbol{W} = \begin{bmatrix} 20 & 19 & 1 \\ 1 & 22 & 21 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ e $\boldsymbol{k} = \begin{bmatrix} 42 \\ 62 \\ 0 \\ 1 \\ 0 \end{bmatrix}$.

La sequenza di valori di τ_s che vengono calcolati ad ogni step sono rispettivamente:

$$\tau_{s=1} = 0, \tau_{s=2} = 0, \tau_{s=3} = 1, \tau_{s=4} = 1, \tau_{s=5} = 1, \tau_{s=6} = 1.$$

Questi valori indicano, in valore assoluto, i termini che costituiscono la matrice $\mathbf{LU}$ e saranno assegnati alla prima riga o alla seconda riga di tale matrice seguendo l'ordine dell'elemento $\bar{j}_s$ delle seguenti coppie:

$$(\bar{i}_{s=1}, \bar{j}_{s=1}) = (3, 1), (\bar{i}_{s=2}, \bar{j}_{s=2}) = (5, 3), (\bar{i}_{s=3}, \bar{j}_{s=3}) = (4, 2),$$

$$(\bar{i}_{s=4}, \bar{j}_{s=4}) = (1, 1), (\bar{i}_{s=5}, \bar{j}_{s=5}) = (1, 2), (\bar{i}_{s=6}, \bar{j}_{s=6}) = (2, 3).$$

La matrice risultante è $\mathbf{LU} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$.

Si riassume questa prima implementazione nel seguente modo:

$s = 1$

$\tau_1 = min\{\lfloor\frac{42}{40}\rfloor = \lfloor\frac{62}{44}\rfloor = \lfloor\frac{0}{1}\rfloor = \lfloor\frac{1}{1}\rfloor = \lfloor\frac{0}{1}\rfloor\} = 0,$

$J_1 = \{(3,1),(5,3)\}$, si sceglie $(3,1)$

$\boldsymbol{w}_1^1 = \begin{bmatrix} 20 & 19 & 1 \end{bmatrix}, k_1^1 = 42, \qquad \boldsymbol{w}_2^1 = \begin{bmatrix} 1 & 22 & 21 \end{bmatrix}, k_2^1 = 62,$

$\boldsymbol{w}_3^1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^1 = 0, \qquad \boldsymbol{w}_4^1 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}, k_4^1 = 1,$

$\boldsymbol{w}_5^1 = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}, k_5^1 = 0,$

$\boldsymbol{u}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \boldsymbol{l}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$

$s = 2$

$\tau_2 = min\{\lfloor\frac{42}{40}\rfloor = \lfloor\frac{62}{44}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{1}{1}\rfloor = \lfloor\frac{0}{1}\rfloor\} = 0,$

$J_2 = \{(4,2),(5,3)\}$, si sceglie $(5,3)$

$\boldsymbol{w}_1^2 = \begin{bmatrix} 20 & 19 & 1 \end{bmatrix}, k_1^2 = 42, \qquad \boldsymbol{w}_2^2 = \begin{bmatrix} 1 & 22 & 21 \end{bmatrix}, k_2^2 = 62,$

$\boldsymbol{w}_3^2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^2 = 0, \qquad \boldsymbol{w}_4^2 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}, k_4^2 = 1,$

$\boldsymbol{w}_5^2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_5^2 = 0,$

$\boldsymbol{u}_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \boldsymbol{l}_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$

$s = 3$

$\tau_3 = min\{\lfloor\frac{42}{40}\rfloor = \lfloor\frac{62}{44}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{1}{1}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 1,$

$J_3 = \{(4,2)\}$, si sceglie $(4,2)$

$\boldsymbol{w}_1^3 = \begin{bmatrix} 20 & 19 & 1 \end{bmatrix}, k_1^3 = 42, \qquad \boldsymbol{w}_2^3 = \begin{bmatrix} 1 & 22 & 21 \end{bmatrix}, k_2^3 = 62,$

$\boldsymbol{w}_3^3 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^3 = 0, \qquad \boldsymbol{w}_4^3 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_4^3 = 0,$

$\boldsymbol{w}_5^3 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_5^3 = 0,$

$\boldsymbol{u}_3 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \boldsymbol{l}_3 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}.$

$s = 4$

$\tau_4 = min\{\lfloor\frac{42}{40}\rfloor = \lfloor\frac{62}{44}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 1,$

$J_4 = \{(1,1)\}$, si sceglie $(1,1)$

$\boldsymbol{w}_1^4 = \begin{bmatrix} 0 & 19 & 1 \end{bmatrix}, k_1^4 = 22, \qquad \boldsymbol{w}_2^4 = \begin{bmatrix} 0 & 22 & 21 \end{bmatrix}, k_2^4 = 61,$

$\boldsymbol{w}_3^4 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^4 = 0, \qquad \boldsymbol{w}_4^4 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_4^4 = 0,$

$\boldsymbol{w}_5^4 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_5^4 = 0,$

$\boldsymbol{u}_4 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \boldsymbol{l}_4 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}.$

$s = 5$

$\tau_5 = min\{\lfloor\frac{22}{20}\rfloor = \lfloor\frac{61}{43}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 1,$

$J_5 = \{(1,2)\}$, si sceglie $(1,2)$

$\boldsymbol{w}_1^5 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, k_1^5 = 3, \qquad \boldsymbol{w}_2^5 = \begin{bmatrix} 0 & 0 & 21 \end{bmatrix}, k_2^5 = 39,$

$\boldsymbol{w}_3^5 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^5 = 0, \qquad \boldsymbol{w}_4^5 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_4^5 = 0,$

$\boldsymbol{w}_5^5 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_5^5 = 0,$

$\boldsymbol{u}_5 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \boldsymbol{l}_5 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}.$

$s = 6$

$\tau_6 = min\{\lfloor\frac{3}{1}\rfloor = \lfloor\frac{39}{21}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor = \lfloor\frac{0}{0}\rfloor\} = 1,$

$J_6 = \{(2,3)\}$, si sceglie $(2,3)$

$\boldsymbol{w}_1^6 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_1^6 = 2, \qquad \boldsymbol{w}_2^6 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_2^4 = 18,$

$\boldsymbol{w}_3^6 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_3^6 = 0, \qquad \boldsymbol{w}_4^6 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_4^6 = 0,$

$\boldsymbol{w}_5^6 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, k_5^6 = 0,$

$\boldsymbol{u}_6 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \boldsymbol{l}_6 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}.$

La sequenza dei lati τ afferma che i risultati ottenuti sono massimali: infatti i valori τ trovati a ogni step sono crescenti e diventano costanti al termine della sequenza, cioè negli ultimi step si trova sempre lo stesso valore.

A partire da questi risultati si procede all'applicazione dell'Algoritmo HMB per confermare i risultati già trovati, partendo dalla considerazione che l'ipercubo è già massimo. Questo algoritmo è composto da ulteriori 2 step e cerca di massimizzare o minimizzare i posti indicati da $\bar{j}_s = (1, 3)$.

I fattori di incremento e decremento valutati ai diversi step, per ogni elemento $\bar{j}_s$, come ci si poteva aspettare sono nulli; questo a conferma del fatto che i risultati trovati con l'Algoritmo HB sono massimali, come suggerito dalla condizione (3.17).

La matrice dei boundaries al termine dell'applicazione di questo secondo algoritmo è uguale alla precedente: $\mathbf{LU} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$.

Riportando queste coordinate nel sistema originale, con la funzione *trasla_LU*, si trova *LU_def* $= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$.

Questa matrice individua l'ipercubo massimale incluso in $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

5.2.2 Conteggio delle marcature

All'interno dell'ipercubo intero massimale determinato con l'Algoritmo HB e definito da *LU_def*, si conta un numero di marcature $\boldsymbol{M}$ pari a 12. Nello spazio delimitato dalle GMEC invece le marcature $\boldsymbol{M}$ presenti sono 29.

Si nota che come previsto l'ipercubo è sempre maggiormente selettivo quanto più aumenta il numero di posti, infatti elimina un gran numero di marcature rispetto allo spazio delimitato dalle GMEC.

Si può pensare di stabilire un criterio con cui varia tale riduzione, ma ciò non segue nessun tipo di andamento, essendo variabile con il numero di posti e quindi con le coordinate dell'ipercubo.

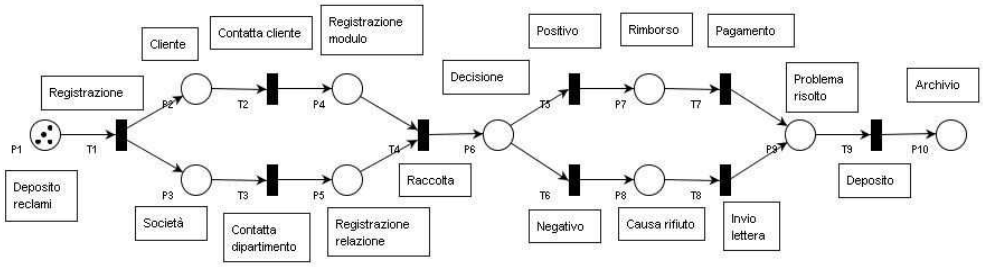

Figura 5.1: Rete di Petri per il processo di gestione dei reclami

5.3 Caso 3

Si consideri un processo per la gestione dei reclami.

Un reclamo viene dapprima registrato, successivamente vengono contattati sia il cliente che ha effettuato il reclamo che il dipartimento interessato dal reclamo. Al cliente si richiede di compilare un modulo con tutte le informazioni necessarie, mentre al dipartimento viene richiesta una relazione per ottenere maggiori informazioni. Queste richieste possono essere portate avanti in parallelo. Successivamente tutte le informazioni sono raccolte e valutate al fine di prendere una decisione. In base alla decisione presa viene inviato un risarcimento al cliente oppure una lettera con le motivazioni del mancato risarcimento. Infine il reclamo viene depositato.

Il processo descritto viene modellato con una rete di Petri P/T dando un significato fisico a ciascun posto che rappresenta uno stato in cui il reclamo può trovarsi, e a ciascuna transizione che rappresenta un evento.

La rete di Petri che descrive questo processo è rappresentata in Figura (5.1).

Si consideri il seguente insieme di GMEC

$$\mathcal{M}(\boldsymbol{W},\boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^{10} \mid$$
$$3m(p_1) + m(p_2) + m(p_3) + 2m(p_4) + 2m(p_5) + m(p_7) + 2m(p_9) + m(p_{10})$$
$$\leq 25,$$
$$2m(p_2) + 3m(p_3) + m(p_4) + m(p_5) + 2m(p_7) + 4m(p_8) + m(p_9) + 2m(p_{10})$$
$$\leq 32,$$
$$m(p_1) + m(p_3) + 3m(p_4) + 2m(p_5) + 2m(p_6) + 2m(p_7) + m(p_8) + 3m(p_{10})$$
$$\leq 23,$$
$$2m(p_1) + 2m(p_2) + m(p_4) + 3m(p_5) + 2m(p_6) + m(p_7) + m(p_9) + m(p_{10})$$
$$\leq 20\}.$$

Si determinino le coordinate dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W},\boldsymbol{k})$, gli insieme $\mathcal{M}(\boldsymbol{W},\boldsymbol{k})$ di marcature legali decentralizzate considerando una suddivisione dei posti di tipo non singleton e il numero di marcature presenti nell'ipercubo massimale.

5.3.1 Determinazione delle coordinate dell'ipercubo

L'applicazione dell'Algoritmo HB richiede la conoscenza dei risultati del problema di programmazione lineare intero per determinare i dati elaborati dall'algoritmo.

Il LIPP fornisce *X_opt* $= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix}^T$ e *F_opt* $= 1$.

Con queste informazioni, nota la matrice *W_in* e il vettore *k_in*, per mezzo della funzione *data_input* si trova che i dati elaborati al fine di calcolare i boundaries

dell'ipercubo intero sono:

$$\boldsymbol{W} = \begin{bmatrix} 3 & 1 & 1 & 2 & 2 & 0 & 1 & 0 & 2 & 1 \\ 0 & 2 & 3 & 1 & 1 & 0 & 2 & 4 & 1 & 2 \\ 1 & 0 & 1 & 3 & 2 & 2 & 2 & 1 & 0 & 3 \\ 2 & 2 & 0 & 1 & 3 & 2 & 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad e \quad \boldsymbol{k} = \begin{bmatrix} 21 \\ 27 \\ 15 \\ 15 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}.$$

Questo sistema è costituito da 10 posti, perciò l'algoritmo avrà in totale 20 diversi valori di τ. La sequenza di valori di τ calcolati ai diversi step è:

$$\tau_{s=1} = 0, \tau_{s=2} = 0, \tau_{s=3} = 0, \tau_{s=4} = 0, \tau_{s=5} = 0, \tau_{s=6} = 0, \tau_{s=7} = 0,$$

$$\tau_{s=8} = 0, \tau_{s=9} = 1, \tau_{s=10} = 1, \tau_{s=11} = 1, \tau_{s=12} = 1, \tau_{s=13} = 1, \tau_{s=14} = 1,$$

$$\tau_{s=15} = 1, \tau_{s=16} = 1, \tau_{s=17} = 1, \tau_{s=18} = 1, \tau_{s=19} = 2, \tau_{s=20} = 3.$$

La sequenza di coppie di indici $(\bar{i}_s, \bar{j}_s)$ scelte a ogni step sono, in sequenza:

$$(5,1), (6,2), (7,3), (8,4), (10,6), (11,7), (12,8), (13,9), (3,1), (3,3), (3,4),$$

$$(3,5), (9,5), (3,6), (3,7), (3,8), (3,10), (4,2), (14,10), (4,9).$$

Conoscendo questi elementi, l'algoritmo determina la matrice $\mathbf{LU}$ così definita:

$$\mathbf{LU} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 1 \end{bmatrix}.$$

Non si riporta in questa sezione un riassunto dei parametri calcolati a ogni step, a causa della complessità dell'esercizio che presenta 10 posti associati quindi a un algoritmo di 20 step.

La sequenza di valori di τ indica che l'Algoritmo HMB avrà 19 step, poichè $\mu = 20$. Seguendo l'ordine indicato dai valori di $\bar{j}_s$ si decide quali sono i primi $\mu - 1$ termini di $\mathbf{LU}$ candidati a subire un incremento o un decremento. La quantità da sommare è sempre nulla a ogni step, quindi non vi è nessuna modifica nei valori dei boundaries, per cui anche questo algoritmo conferma quanto già trovato dall'Algoritmo HB, $\mathbf{LU} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 1 \end{bmatrix}$.

5.3.2 Determinazione delle marcature decentralizzate

In questo esercizio si considera una suddivisione di posti non singleton. Tale scelta, differente dalle precedenti, è giustificata dall'esigenza di poter testare la funzionalità degli algoritmi implementati in modi differenti; ciò significa che, se nei casi precedenti, considerando posti singleton, si è verificato che gli algoritmi funzionano correttamente e non presentano alcuna difficoltà di esecuzione, ora si vuole accertare che anche con una suddivisione di posti di tipo non singleton, gli algoritmi conservino le stesse caratteristiche e funzionalità.

In questo caso la matrice $\boldsymbol{P}$ che rappresenta la suddivisione dei posti è:

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

L'esercizio considera un insieme di 6 posti P_i, ognuno dei quali contiene i posti p_i, perciò la matrice $\boldsymbol{P}$ ha 6 righe e $m = 10$ colonne, poichè ogni posto p_i deve essere presente in un insieme P_i.

La matrice *W_in* in ingresso è una matrice 4×10. Ci si aspetta che la matrice *W_out* abbia dimensione 24×10 e il vettore colonna *k_out* abbia dimensione 24×1.

Con la funzione *trova_W_out_k_out* si ottiene la matrice *W_out* e successivamente alla determinazione di questa matrice si può calcolare il vettore *k_out*, ottenendo:

$$W\_out = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad e \quad k\_out = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 0 \\ 1 \\ 7 \\ 0 \\ 5 \\ 2 \\ 0 \\ 6 \\ 5 \\ 1 \\ 1 \\ 5 \\ 2 \\ 3 \\ 3 \\ 2 \\ 2 \\ 4 \\ 2 \\ 1 \\ 4 \end{bmatrix}.$$

La moltiplicazione della matrice *W_out* per il vettore $\boldsymbol{m}$ trova gli insiemi di marcature massimali decentralizzate scritte per esteso. Il risultato di tale operazione

è:

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} m(p_1) \\ m(p_2) \\ m(p_3) \\ m(p_4) \\ m(p_5) \\ m(p_6) \\ m(p_7) \\ m(p_8) \\ m(p_9) \\ m(p_{10}) \\ m(p_{11}) \\ m(p_{12}) \\ m(p_{13}) \\ m(p_{14}) \\ m(p_{15}) \\ m(p_{16}) \\ m(p_{17}) \\ m(p_{18}) \\ m(p_{19}) \\ m(p_{20}) \\ m(p_{21}) \\ m(p_{22}) \\ m(p_{23}) \\ m(p_{24}) \end{bmatrix} \leq \begin{bmatrix} 3 \\ 2 \\ 4 \\ 0 \\ 1 \\ 7 \\ 0 \\ 5 \\ 2 \\ 0 \\ 6 \\ 5 \\ 1 \\ 1 \\ 5 \\ 2 \\ 3 \\ 3 \\ 2 \\ 2 \\ 4 \\ 2 \\ 1 \\ 4 \end{bmatrix}.$$

Al termine di questo calcolo, la funzione *trasla_LU* determina le coordinate dell'ipercubo intero massimale nel sistema originale e si raggiunge il seguente risultato:

$$\mathbf{LU} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 3 & 3 \end{bmatrix}.$$

Si noti che nel definire l'insieme di GMEC non sono stati scelti vincoli con preciso significato fisico, ma si è scelto di imporre alla rete un certo quantitativo di marche. Inoltre ogni GMEC non imponeva dei vincoli su ciascun posto presente nella rete; ciò significa che i posti non presenti in una GMEC devono essere

presenti in almeno un'altra GMEC, al fine di essere vincolati e di non assumere valori infiniti.

Come anticipato, lo scopo di tale esercizio è quello di testare la funzionalità degli algoritmi implementati su un esempio numericamente più complesso. Infatti la presenza di più vincoli, cioè di più GMEC singole $(\boldsymbol{w}, k)$ che costituiscono la GMEC multipla $(\boldsymbol{W}, \boldsymbol{k})$, la presenza di un elevato numero di posti oltre a una suddivisione non singleton di questi ultimi, aumentano le difficoltà di esecuzione dei vari passi ma allo stesso tempo rendono l'esercizio più completo al fine di testare tutte le varianti possibili.

I risultati ottenuti con l'Algoritmo HMB portano alle stesse conclusioni a cui conduce l'Algoritmo HB, cioè che l'ipercubo ottenuto con l'applicazione del primo algoritmo è massimale.

5.3.3 Conteggio delle marcature

In questo caso, dato l'elevato numero di posti presenti nella rete, il conteggio delle marcature è un'operazione più complicata rispetto agli altri due casi precedenti, nei quali l'enumerazione avveniva in uno spazio rispettivamente di 2 dimensioni o di 3 dimensioni. Ora ci si trova a dover far fronte all'esigenza di eseguire un conteggio all'interno di uno spazio che non si può rappresentare, infatti si sviluppa in 10 dimensioni, quindi ci si limita esclusivamente a eseguire il conteggio dei vettori costituiti di soli elementi positivi.

Le marcature $\boldsymbol{M}$ presenti all'interno dell'ipercubo delimitato dai boundaries sono 6144.

Come fatto precedentemente, è di interesse il confronto con il numero di marcature presenti all'interno dello spazio delimitato dalle GMEC. La durata dell'elaborazione richiede tempi molti più lunghi rispetto agli altri file eseguiti a causa del notevole peso computazionale derivato dall'assegnazione di una rete di Petri con 10 posti in ingresso e alla possibilità che in alcuni casi i posti non siano limitati.

5.4 Caso 4

Si consideri il sistema in Figura (5.2).

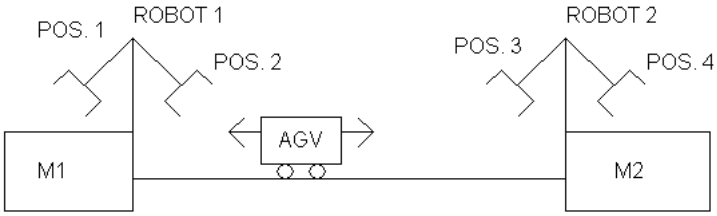

Figura 5.2: Funzionamento di un sistema produttivo

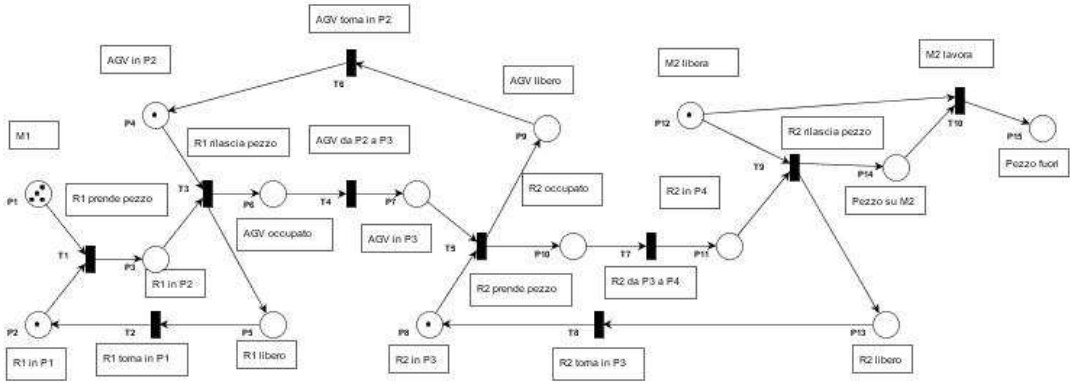

Figura 5.3: Rete di Petri per un sistema robot

Il robot 1 (R1), se in posizione 1 (P1) afferra un pezzo dal magazzino M1, supposto illimitato, si porta nella posizione 2 (P2) e deposita il pezzo sull'AGV, se quest'ultimo è presente, quindi torna in posizione 1 (P1) per prendere un altro pezzo dal magazzino M1. Una volta carico, l'AGV si sposta dalla posizione 2 (P2) alla posizione 3 (P3). Il robot 2 (R2), se in posizione 3 (P3), preleva il pezzo presente sull'AGV, si porta in posizione 4 (P4) e lo deposita sulla macchina M2 se questa è libera, quindi torna in posizione 3 (P3). Quando il robot 2 (R2) effettua la presa del pezzo sull'AGV, quest'ultimo torna nella posizione 2 (P2) in attesa di un nuovo pezzo da M1. In seguito alla lavorazione in M2, il pezzo esce dal sistema e M2 torna libera per accogliere un nuovo pezzo.

Il funzionamento di questo sistema viene descritto con una rete di Petri modellata come nella Figura (5.3).

Si consideri il seguente insieme di GMEC:

$$\mathcal{M}(\boldsymbol{W},\boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^{15} \mid$$
$$m(p_1) + 3m(p_3) + 2m(p_4) + 4m(p_5) + m(p_6) + 3m(p_8) + 2m(p_9) + m(p_{10}) +$$
$$2m(p_{11}) + 2m(p_{13}) + m(p_{14}) + 4m(p_{15}) \leq 44,$$
$$2m(p_1) + m(p_2) + 2m(p_3) + 2m(p_5) + m(p_6) + m(p_7) + 3m(p_9) + 5m(p_{11}) +$$
$$2m(p_{12}) + m(p_{13}) + 3m(p_{14}) + m(p_{15}) \leq 39,$$
$$2m(p_1) + 4m(p_3) + m(p_4) + 2m(p_5) + 2m(p_7) + m(p_8) + 4m(p_9) + 2m(p_{10}) +$$
$$m(p_{11}) + 3m(p_{12}) + m(p_{15})) \leq 37,$$
$$m(p_1) + 3m(p_2) + 2m(p_4) + m(p_5) + m(p_6) + 3m(p_7) + 2m(p_8) + 2m(p_{10}) +$$
$$m(p_{11}) + m(p_{12}) + 3m(p_{13}) + 3m(p_{14}) \leq 45,$$
$$2m(p_1) + 2m(p_2) + 4m(p_3) + 3m(p_4) + m(p_5) + 2m(p_6) + m(p_7) + 3m(p_9) +$$
$$2m(p_{10}) + m(p_{12}) + m(p_{14}) + 2m(p_{15}) \leq 51\}.$$

Si determinino le coordinate dell'ipercubo intero massimale in $\mathcal{M}(\boldsymbol{W},\boldsymbol{k})$, gli insieme $\mathcal{M}(\boldsymbol{W},\boldsymbol{k})$ di marcature legali decentralizzate considerando una suddivisione di posti di tipo non singleton e il numero di marcature presenti nell'ipercubo massimale.

5.4.1 Determinazione delle coordinate dell'ipercubo

I risultati del problema di programmazione lineare sono:
$X\_opt = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$ e $F\_opt = 1$.

Conoscendo la matrice *W_in* e il vettore *k_in* che definiscono le GMEC, usando la funzione *data_input*, si determinano i dati in ingresso all'Algoritmo HB:

$$\boldsymbol{W} = \begin{bmatrix}
1 & 0 & 3 & 2 & 4 & 1 & 0 & 3 & 2 & 1 & 2 & 0 & 2 & 1 & 4 \\
2 & 1 & 2 & 0 & 2 & 1 & 1 & 0 & 3 & 0 & 5 & 2 & 1 & 3 & 1 \\
2 & 0 & 4 & 1 & 2 & 0 & 2 & 1 & 4 & 2 & 1 & 3 & 0 & 0 & 1 \\
1 & 3 & 0 & 2 & 1 & 1 & 3 & 2 & 0 & 2 & 1 & 1 & 3 & 3 & 0 \\
2 & 2 & 4 & 3 & 1 & 2 & 1 & 0 & 3 & 2 & 0 & 1 & 0 & 1 & 2 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix},$$

$$\boldsymbol{k} = \begin{bmatrix} 37 & 28 & 23 & 44 & 40 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\boldsymbol{T}}.$$

Questo algoritmo è costituito da 30 step. Si elencano i valori di τ_s ai diversi step:

$$(0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,2,2,3,3).$$

Le coppie $(\bar{i}_s, \bar{j}_s)$ per ogni step sono:

$$(7,2),(8,3),(9,4),(10,5),(11,6),(12,7),(13,8),(15,10),(16,11),(17,12),$$

$$(18,13),(19,14),(20,15),(3,1),(6,1),(3,3),(3,4),(3,5),(3,7),(3,8),(3,9),$$

$$(3,10),(3,11),(3,12),(3,15),(2,14),(2,2),(2,6),(14,9),(2,13).$$

Seguendo la sequenza di valori $\bar{j}_s$ si stabiliscono quali posti possono aumentare o diminuire il proprio bound.

Si riporta la matrice $\mathbf{LU}$ ottenuta al termine dell'esecuzione

$$\mathbf{LU} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 3 & 1 & 1 \end{bmatrix}.$$

Come per l'esercizio precedente la mole di dati elevata suggerisce di porre l'attenzione direttamente sui risultati dell'elaborazione.

I valori di τ_s permettono di individuare l'indice μ, il cui valore, diminuito di uno, rappresenta il numero di step eseguito nell'Algoritmo HMB per accertarsi se i risultati trovati sono massimali o in caso contrario per determinarli.

Anche in questo esercizio, come nei precedenti, la quantità di cui incrementare o decrementare è nulla. Quindi nessun posto subirà un cambiamento e i boundaries saranno gli stessi calcolati precedentemente. Questo conferma che la matrice $\mathbf{LU}$ resta uguale all'algoritmo precedente, ovvero

$$\mathbf{LU} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 3 & 1 & 1 \end{bmatrix}.$$

5.4.2 Determinazione delle marcature decentralizzate

Questo esercizio presenta caratteristiche analoghe al precedente caso trattato, ma con l'ulteriore aggiunta di posti nella rete di Petri. La suddivisione di posti che si considera è anche stavolta di tipo non singleton.

In questo esercizio si massimizza ulteriormente la quantità di dati in ingresso, numero di posti e GMEC singole, per dimostrare come gli algoritmi proposti in questa tesi, possano condurre a un risultato esaustivo per qualunque tipo di applicazione, sia essa un sistema fisico con un elevato numero di posti o un esempio didattico con un numero limitato di posti.

L'insieme dei posti è stato suddiviso in 4 insiemi P_i ognuno contenente i posti p_i,

dunque la matrice $\boldsymbol{P}$ che rappresenta la suddivisione dei posti è:

$$\boldsymbol{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

La matrice *W_in* è una matrice 5×15, la matrice $\boldsymbol{P}$ è una matrice 4×15, perciò la matrice *W_out* rappresentante le GMEC decentralizzate ha dimensione 20×15, allo stesso modo il vettore *k_out* ha dimensione 20×1.

La funzione *trova_W_out_k_out* determina

$$W\_out = \begin{bmatrix} 1 & 0 & 3 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 3 & 2 & 4 & 0 & 0 & 3 & 2 & 1 & 2 & 0 & 2 & 0 & 0 \\ 1 & 0 & 3 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\ 2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 \\ 2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 \\ 2 & 0 & 4 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 1 & 2 & 0 & 0 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 3 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & 1 & 0 & 3 & 0 & 0 \\ 1 & 3 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}, \qquad k\_out = \begin{bmatrix} 10 \\ 2 \\ 24 \\ 15 \\ 8 \\ 3 \\ 19 \\ 12 \\ 9 \\ 2 \\ 17 \\ 10 \\ 10 \\ 5 \\ 24 \\ 13 \\ 14 \\ 5 \\ 19 \\ 17 \end{bmatrix}.$$

L'insieme di GMEC decentralizzato è così definito:

$$\begin{bmatrix}
1 & 0 & 3 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 3 & 2 & 4 & 0 & 0 & 3 & 2 & 1 & 2 & 0 & 2 & 0 & 0 \\
1 & 0 & 3 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\
2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 \\
2 & 1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 \\
2 & 0 & 4 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 4 & 1 & 2 & 0 & 0 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 4 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 3 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & 1 & 0 & 3 & 0 & 0 \\
1 & 3 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2
\end{bmatrix} \times \begin{bmatrix}
m(p_1) \\ m(p_2) \\ m(p_3) \\ m(p_4) \\ m(p_5) \\ m(p_6) \\ m(p_7) \\ m(p_8) \\ m(p_9) \\ m(p_{10}) \\ m(p_{11}) \\ m(p_{12}) \\ m(p_{13}) \\ m(p_{14}) \\ m(p_{15}) \\ m(p_{16}) \\ m(p_{17}) \\ m(p_{18}) \\ m(p_{19}) \\ m(p_{20})
\end{bmatrix} \leq \begin{bmatrix}
10 \\ 2 \\ 24 \\ 15 \\ 8 \\ 3 \\ 19 \\ 12 \\ 9 \\ 2 \\ 17 \\ 10 \\ 10 \\ 5 \\ 24 \\ 13 \\ 14 \\ 5 \\ 19 \\ 17
\end{bmatrix}.$$

La funzione *trasla_LU* fornisce il seguente risultato

$$\mathbf{LU} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 4 & 1 & 1 & 1 & 3 & 1 & 1
\end{bmatrix}.$$

che rappresenta le coordinate dell'ipercubo intero massimale interno allo spazio delimitato dall'insieme di GMEC.

Alle GMEC utilizzate per descrivere la rete non è stato associato nessun significato preciso, mentre la suddivisione dei posti ha seguito un particolare criterio per il quale ogni insieme è costituito dal sottoinsieme di posti che lavorano nello stesso ambito. Infatti, il primo insieme P_1 contiene i posti p_1, p_2, p_3, p_4, p_5, i quali fanno riferimento all'utilizzo della macchina M1 e delle operazioni eseguite dal robot R1. Il secondo insieme P_2 contiene i posti p_6, p_7 i quali descrivono le operazioni svolte dall'AGV. Il terzo insieme P_3 è costituito dai posti $p_8, p_9, p_{10}, p_{11}, p_{13}$ e

esprimono l'utilizzo del robot R2, che passa dalla posizione P3 alla posizione P4. Infine, l'ultimo insieme è P_4 che comprende i posti p_{12}, p_{14}, p_{15} i quali raffigurano lo stato della macchina M2.

Si sottolinea che anche in questo caso ogni posto è stato vincolato in almeno una GMEC. Se così non fosse stato, la possibilità che un posto potesse assumere valore infinito, non essendo mai limitato, avrebbe reso impossibile l'applicazione degli algoritmi senza modifiche sostanziali che limitino a priori il numero massimo di marcature presenti in ogni posto a prescindere dalle GMEC.

5.4.3 Conteggio delle marcature

Il numero di marcature $\boldsymbol{M}$ all'interno dell'ipercubo intero definito dai boundaries è 552960.

Sarebbe interessante effettuare un confronto tra il numero di marcature presenti all'interno dell'ipercubo massimale e le marcature presenti all'interno dello spazio delimitato dalle GMEC, per confermare che quanto più aumenta il numero di posti, e quindi quanto più aumenta la possibilità di avere un conteggio maggiore, tanto più restrittivo risulta l'ipercubo rispetto allo spazio definito dalle GMEC: infatti l'ipercubo racchiuderà al suo interno una porzione sempre più piccola di marcature.

Il primo conteggio, poichè fa riferimento a uno spazio maggiormente limitato, non comporta alcun'osservazione sul tempo impiegato per l'esecuzione. Infatti non si osserva un'eccessiva attesa prima di giungere al risultato. Invece il secondo conteggio potrebbe non giungere mai a un risultato finale. Ciò è motivato dal grande peso computazionale dovuto all'elevato numero di posti ma anche alla definizione delle GMEC che potrebbe rendere illimitato il valore massimo assunto da un posto.

5.5 Caso parametrico

In questo paragrafo l'attenzione è rivolta ai tempi impiegati dal software sviluppato per l'esecuzione del programma. A tale scopo è utile considerare sempre la stessa struttura di rete per ogni caso, per non inserire dei cambiamenti che possano

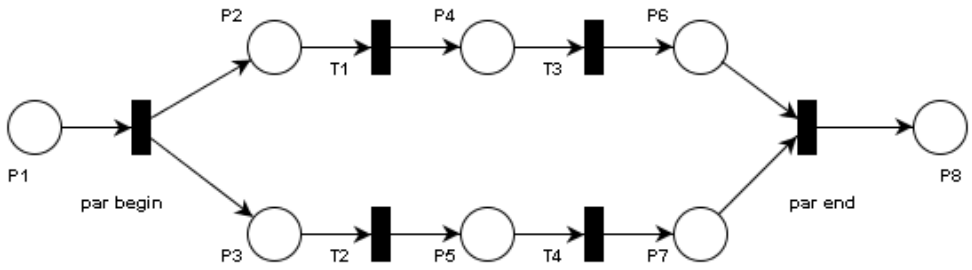

Figura 5.4: Esempio di rete parametrica

incidere in maniera significativa durante la valutazione dei tempi, quindi si sceglie una rete parametrica in cui il numero dei posti varia secondo un preciso criterio.

Si crea tale rete parametrica costituita dalla successione di una transizione *par begin*, una struttura di sequenzialità per ogni ramo della transizione par begin e una transizione *par end*. Si indica con s il numero di eventi abilitati dallo scatto della transizione *par begin* uguale al numero di eventi che devono verificarsi per far scattare la transizione *par end*; con p si indica il numero di posti in sequenza in ogni ramo tra la transizione *par begin* e la transizione *par end*.

In Figura (5.4) è rappresentata una rete in cui s è pari a 2 e p vale 3. In totale la rete è costituita da $(s \times p) + 2$ posti.

Nel definire la rete è stato scelto un criterio parametrico il quale deve essere rispettato anche nella definizione delle GMEC utilizzate per applicare gli algoritmi. Per ogni GMEC applicata alle p coppie (nel caso di Figura (5.4)) di posti interni alla sequenzialità deve essere considerata la somma delle marcature di ogni posto e tale somma si impone minore o uguale a 1; alle p GMEC riguardanti i posti all'interno della sequenzialità, si aggiungono altre 2 GMEC che vincolano in maniera scorrelata le marcature dei due posti esterni. In totale si avranno dunque $p + 2$ GMEC che assicurano la presenza contemporanea nella rete di un numero massimo di $p + 2$ marcature.

Per la rete in Figura (5.4), l'insieme di GMEC è il seguente:

$$\begin{aligned}&\mathcal{M}(\boldsymbol{W},\boldsymbol{k}) = \{\boldsymbol{m} \in \mathbb{N}^8 \mid \\ &m(p_1) \leq 1, \\ &m(p_2) + m(p_3) \leq 1, \\ &m(p_4) + m(p_5) \leq 1, \\ &m(p_6) + m(p_7) \leq 1, \\ &m(p_8) \leq 1\}.\end{aligned}$$

Definita la rete e le GMEC, si eseguono i test che indicano quanto tempo occorre per avere un risultato. Per eseguire i test sono stati scelti dei valori arbitrari per s e per p, in base ai quali si ha una rete più o meno complicata e quindi tempi più o meno lunghi. E' stato fissato il parametro s pari a 2, 6 e 9, mentre p pari a 2, 3 e 5, ottenendo delle reti che variano da 6 posti a 47 posti.

I test sono stati effettuati su un computer con CPU Pentium 4 a 2.8 GHz e RAM di 256 MB.

In tabella vengono riportati i tempi, espressi in secondi, spesi per l'esecuzione dell'Algoritmo HMB.

$\boldsymbol{p} \setminus \boldsymbol{s}$	2	6	9
2	0.272335	0.839425	0.997365
3	0.329425	0.887091	1.873017
5	0.603750	1.950972	4.779424

Come si nota dai valori riportati, tanto più aumenta il numero di posti nella rete tanto più il tempo d'esecuzione cresce. Ciò era prevedibile poichè il numero di step di cui si compone l'algoritmo è direttamente proporzionale a m ovvero al numero di posti totali nella rete, ma soprattutto, più sono i posti nella rete, tanto più grandi sono le matrici in ingresso al programma e quindi ci sarà un rallentamento dovuto al maggior numero di calcoli realizzati.

Inoltre si è voluto effettuare un altro test che calcola il tempo per l'esecuzione di un caso in cui i posti sono suddivisi in modo non singleton. Si precisa che la matrice $\boldsymbol{P}$ che contiene la suddivisione dei posti è definita come la matrice *W_in* in ingresso, infatti si considerano tanti insiemi di posti P_i quante sono le GMEC che definiscono il problema e ogni insieme contiene solo i posti p_i presenti nella GMEC.

Nella seguente tabella si riportano i tempi di svolgimento del programma.

$p \setminus s$	2	6	9
2	0.572660	1.571516	1.970390
3	0.698623	1.934852	3.636547
5	1.321578	3.924730	9.786135

Come ci si aspettava, i tempi nel caso non singleton sono superiori rispetto al caso precedente. In particolare si nota che i tempi del secondo caso sono circa il doppio dei tempi precedentemente ottenuti. Ciò può essere attribuito al fatto che, per come è definita la matrice $\boldsymbol{P}$, delle stesse dimensioni della matrice *W_in* di definizione delle GMEC, il programma deve affrontare, in aggiunta ai calcoli per determinare i boundaries, una mole simile di calcoli che porta a un raddoppio dei tempi di elaborazione.

Infine è di interesse fare una valutazione dei tempi impiegati per contare le marcature all'interno dell'ipercubo massimale determinato con gli algoritmi studiati. Avere una stima dei tempi impiegati per questo conteggio è particolarmente importante perchè, oltre a essere un aspetto nuovo rispetto a quanto è presente in letteratura, conduce sempre a un risultato finito; al contrario un conteggio all'interno dello spazio delimitato dalle GMEC potrebbe portare a un'elaborazione infinita senza mai trovare un risultato.

In tabella si riportano i tempi, espressi in secondi, per calcolare le marcature all'interno dell'ipercubo massimale.

$p \setminus s$	2	6	9
2	0.765561	1.725765	2.375036
3	0.849274	2.329903	3.874122
5	1.421067	4.295428	10.418774

I tempi forniti dalle elaborazioni sono conformi a quanto ottenuto nei precedenti casi, infatti vi è un lievo aumento del tempo d'esecuzione del programma per ciascun caso, riconducibile a un'ulteriore verifica, eseguita analizzando la matrice *W_in* e il vettore *k_in*, per appurare che le marcature conteggiate rispettino i vincoli imposti dalla GMEC globale $\mathcal{M}(\boldsymbol{W}, \boldsymbol{k})$.

Capitolo 6

Conclusioni

In questa tesi è stato affrontato il problema del controllo decentralizzato utilizzando le reti di Petri. Si è esposto l'approccio proposto da Basile, Giua e Seatzu [1], rivolgendo maggiore interesse alla soluzione proposta che adotta un problema di programmazione lineare intero e l'applicazione di due algoritmi denominati Hypercube Bound e Hypercube Maximal Bound. L'implementazione di tali algoritmi mirati a ricercare i boundaries di un ipercubo intero interno a un insieme di GMEC è stata l'obiettivo di questo lavoro di tesi.

Un problema di programmazione lineare intero viene risolto per ricercare il centro $\boldsymbol{c}$ e il lato 2τ dell'ipercubo intero interno all'insieme di GMEC. Durante lo studio del problema si è proposta una variazione del campo di definizione delle variabili del problema: infatti, ora, sia il centro $\boldsymbol{c}$ che il lato τ appartengono all'insieme $\mathbb{Z}$ dei numeri interi, poichè considerare queste variabili appartenenti all'insieme dei numeri reali porta a incongruenze nella determinazione dell'ipercubo.

L'implementazione degli algoritmi, eseguita con Matlab, presenta alcune modifiche che possono considerarsi un miglioramento per l'esecuzione del programma.

Il cambiamento per l'Algoritmo HB riguarda la ricerca dei termini che definiscono la matrice $\boldsymbol{S}$: a ogni step si considera la relazione $j \in L \cup U$ piuttosto che $j \in L_{s-1} \cup U_{s-1}$, la quale permette di calcolare lo stesso numero di elementi $s_{i,j}$ che compongono la matrice $\boldsymbol{S}$ che quindi avrà sempre la stessa dimensione.

Il cambiamento per l'Algoritmo HMB riguarda le variabili utilizzate allo step 3.2; anzichè usare due variabili differenti POS e NEG con valore unitario, essendo queste variabili booleane, si utilizza un'unica variabile POS che assume valore 1

in caso si verifichi la condizione oppure 0 se la condizione non si verifica.

Le funzionalità del software sviluppato con Matlab sono state testate su numerosi esempi con caratteristiche differenti, al fine di dimostrare l'applicabilità a qualsiasi situazione.

Inoltre si è creata una particolare funzione da aggiungere al programma che permette di valutare la bontà della soluzione: infatti questa funzione conta le marcature presenti all'interno dell'ipercubo intero i cui boundaries sono stati calcolati con gli algoritmi proposti. Questo metodo è sicuramente molto efficiente per poter stabilire dei confronti, a parità di condizioni, tra diversi ipercubi poichè un maggior numero di marcature racchiuse all'interno dell'ipercubo conferma la massimalità di quest'ultimo.

Per quel che riguarda il conteggio delle marcature si è provato a stabilire un confronto con le marcature racchiuse all'interno dello spazio delimitato dalle GMEC, senza poter sempre ottenere dei risultati finiti. In particolare si è visto che i problemi in cui le GMEC contenevano dei vincoli per ogni posto presentavano conteggi finiti, mentre i problemi in cui le GMEC non imponeva vincoli per tutti i posti portavano a non concludere mai il conteggio poichè il problema risultava illimitato in almeno una direzione.

Il problema del controllo decentralizzato usando le reti di Petri non è ancora stato ampiamente trattato quindi si pensa che in futuro si possano avere numerosi sviluppi che propongono svariate soluzioni.

Per quel che riguarda l'approccio seguito in questa tesi, si ritiene possa essere interessante stabilire dei criteri per la ripartizione dei posti della rete nel caso si applichi il caso non singleton.

Può avere importanza determinare il numero di marcature racchiuse all'interno dello spazio delimitato dalle GMEC; in tal caso bisognerà stabilire dei criteri che permettano di vincolare l'insieme di GMEC al fine di non avere un problema illimitato.

Un'ultima direzione in cui poter sviluppare l'argomento riguarda la conoscenza dei legami che si creano quando l'applicazione degli algoritmi avviene conoscendo la rete di Petri a cui fa riferimento o il solo insieme di GMEC.

Appendice A

Codice Matlab

A.1 LIPP

A.1.1 Trova_A

```
function [A]=trova_A(W_in)
%trova_A(W_in) permette di determinare
%la matrice A del problema di programmazione
%lineare intero (LIPP).
%*********************************************
%              SINTASSI
%W_in è una matrice di q righe e di
%m colonne; q è il numero di GMEC
%singole, m è il numero totale di posti
%della GMEC.

[q,m]=size(W_in);
%si definisce una variabile UNO che è un
%vettore colonna di uno con m righe.
UNO=ones(m,1);
%al variare di ogni riga della matrice W_in
%tra 1 e q si valuta il vettore F.
for i=1:q
    %si assegna a F lo scalare ottenuto
    %moltiplicando la i-esima riga della
    %matrice W_in per il vettore UNO.
```

```
    F(i,1)=abs(W_in(i,:))*UNO;
end
%la matrice A è ottenuta dalla concatenazione
%della matrice W_in in ingresso e
%del vettore colonna F.
A=[W_in,F];
```

A.2 Algoritmo HM

```
clear all
W_in=[];
k_in=[];
X_opt=[];
[k,W]=data_input(k_in,W_in,X_opt);
A=W;
b=k;
[n_c,m]=size(W);
LU=zeros(2,m);
L=1:m;
U=1:m;

for (s=1:2*m)
tau_s=calcola_tau_s(A,b);
S=calcola_S(W,k,tau_s);
IND=calcola_IND(S);
[LU_out,NEG,POS,L_out,U_out]=calcola_LU(W,tau_s,IND,LU,L,U);
[W_new,k_new]=calcola_W_k(W,k,tau_s,IND,NEG,POS);
LU=LU_out;
L=L_out;
U=U_out;
W=W_new
k=k_new
A=W;
b=k;
end

L_vett=LU(1,:)
U_vett=LU(2,:)
```

A.2.1 Determinazione dei dati in ingresso

```
function [k,W]=data_input(k_in,W_in,X_opt)
%data_input(k_in,W_in,X_opt) determina i
%dati in ingresso all'algoritmo (W e k).
%Alla matrice W_in vengono aggiunte m righe
%relative a m vincoli di non negatività sugli
%m posti, al vettore k_in vengono aggiunti
%m elementi determinati per mezzo di
%un'espressione algebrica.
%***********************************************
%                   SINTASSI
%k_in è un vettore colonna con tante righe
%quante sono le GMEC singole.
%W_in è una matrice con tante righe quante
%sono le GMEC singole e tante
%colonne quanti sono i posti.
%X_opt è un vettore colonna con (m+1) componenti;
%le prime m componenti rappresentano le
%coordinate del centro e l'ultima rappresenta
%il valore massimo di tau_s.

[q,m]=size(W_in);
[q,u]=size(k_in);
[t,u]=size(X_opt);
m=(t-1);
%si definisce C un vettore colonna costituito
%da m componenti uguali alle
%prime m componenti del vettore X_opt.
C=X_opt(1:m,1);
%per i che varia lungo le righe della matrice
%W_in.
 for i=1:q
     %il vettore k è costituito dagli
     %elementi determinati con la seguente
     %espressione.
     k(i,:)=k_in(i)-(W_in(i,:)*C);
 end
%il vettore k in uscita è la concatenazione
%verticale del vettore k trovato con
```

```
%l'espressione precedente e gli elementi
%del vettore colonna C.
k=[k;C];
%si definisce Non_neg una matrice che
%definisce i vincoli di non negatività;
Non_neg=-eye(m,m);
%la matrice W in uscita è costituita dalla
%concatenazione verticale della
%matrice W_in e dalla matrice Non_neg.
W=[W_in;Non_neg];
```

A.2.2 Calcola_tau_s

```
function tau_s=calcola_tau_s(A,b)
%calcola_tau_s(A,b) prende in ingresso la matrice
%A e il vettore colonna b e valuta il rapporto
%tra l'elemento b alla riga i-esima e la somma
%dei valori assoluti degli elementi del vettore
%alla riga i-esima della matrice A.
%Tra tutte le parti intere di tale divisione per
%ogni riga, il valore minimo viene assegnato a tau_s.
%*************************************************
%      SINTASSI
%A è una matrice di n_c righe e di m colonne.
%b è un vettore colonna di n_c righe.

%si assegna a n_c il valore delle righe e a m il
%valore delle colonne della matrice A.
[n_c,m]=size(A);
%per ogni riga da 1 a n_c si valuta tau.
for i=1:n_c
    %si prende la riga i-esima e si esegue il
    %valore assoluto; a è un vettore i cui
    %termini sono i valori assoluti della
    %riga i della matrice A.
    a=abs(A(i,:));
    %sum(a) fornisce la somma di tutti i valori
    %assoluti della riga i-esima; b rappresenta
    %un elemento del vettore colonna in ingresso;
    %si esegue la divisione tra b(i) e sum(a)
    %e si valuta la parte intera.
    tau(i)=floor(b(i)/sum(a));
```

```
end
%tra tutti i tau valutati al variare di i si
%prende quello minore.
tau_s=min(tau);
```

A.2.3 Calcola_S

```
function S=calcola_S(W,k,tau_s)
%calcola_S(W,k,tau_s) consente di
%determinare la matrice S al variare
%dell'indice di riga i e dell'indice di
colonna j. Le righe della matrice
%S saranno sempre pari a i, mentre le
colonne saranno pari a j.
%*********************************************
%                  SINTASSI
%W è una matrice di n_c righe e m colonne.
%k è un vettore colonna di n_c righe.
%tau_s è il valore minimo tra tutti i tau
%calcolati.

%assegno a n_c il valore delle righe della
%matrice W e a m il valore delle
%colonne della matrice W.
[n_c,m]=size(W);
%si assegna a uno un vettore colonna con m
%righe costituito da uno.
uno=ones(m,1);
%al variare di ogni riga tra 1 e n_c e di
%ogni colonna tra 1 e m si valuta
%la matrice S.
for i=1:n_c
    for j=1:m
       %si assegna a w la i-esima riga della
       %matrice W in valore assoluto.
       w=abs(W(i,:));
       %si assegna a w_ij il valore corrispondente
       %al valore assoluto del
       %termine della matrice W che si trova nella
       %i-esima riga e nella j-esima colonna.
       w_ij=abs (W(i,j));
       %si assegna a S un termine a ogni passo
```

```
        %al variare di i e j, valutato
        %per mezzo dell'espressione.
        S(i,j)=((k(i)-tau_s*w*uno)/w_ij);
    end
end
```

A.2.4 Calcola_IND

```
function IND=calcola_IND(S)
%calcola_IND(S) prende in ingresso la
%matrice S e valuta gli indici in cui si
%trovano i valori minimi della matrice
%in ingresso.
%***********************************************
%               SINTASSI
%S è una matrice di n_c righe e m colonne.

%si assegna a minimo il valore minimo della
%matrice S.
minimo=min(min(S));
%assegna a I gli indici di riga dei minimi e
%a E gli indici di colonna dei minimi.
[I,E]=find(S==minimo);
%si assegna a J la concatenazione del vettore I
%degli indici di riga e del vettore E degli
%indici di colonna.
J=[I,E];
%si assegna a IND il primo indice trovato,
%quindi la prima riga della matrice J.
IND=J(1,:);
```

A.2.5 Calcola_LU

```
function [LU_out,NEG,POS,L_out,U_out]=
calcola_LU(W,tau_s,IND,LU,L,U)
%calcola_LU(W,tau_s,IND,LU,L,U) consente
%di determinare lower e upper bound.
```

```
%A ogni passo dell'algoritmo viene valutato
%un termine che va a comporre la matrice LU
%la cui prima riga costituisce il lower bound e
%la seconda riga l'upper bound.
%**************************************************
%                  SINTASSI
%W è una matrice di n_c righe e m colonne.
%tau_s è il valore minimo tra tutti i
%tau valutati.
%IND rappresenta gli indici del primo
%valore minimo trovato nella matrice S.
%LU è inizialmente una matrice di elementi
%nulli e a ogni passo viene definito un termine.
%L è un insieme a cui viene tolto un
%elemento ogni volta che si inserisce un
%valore nella prima riga della matrice LU.
%U è un insieme a cui viene tolto un
%elemento ogni volta che si inserisce un
%valore nella seconda riga della matrice LU.

%si assegna a is il valore dell'indice
%in posizione (1,1) del vettore IND.
is=IND(1,1);
%si assegna a js il valore dell'indice
%in posizione (1,2) del vettore IND.
js=IND(1,2);
%si assegna a n_c il numero di righe
%della matrice W e a m il numero di
%colonne della matrice W.
[n_c,m]=size(W);
LU_out=LU;
 %si valuta l'elemento della matrice W
 %in posizione (is,js).
 if W(is,js)<0
     %se l'elemento è minore di zero
     %allora si inserisce un elemento pari
     %a -tau_s nella prima riga della matrice LU;
     %l'elemento andrà a costituire il vettore
     %del lower bound.
     LU_out(1,js)=-tau_s;
     NEG=1;
     POS=0;
     L_out=trova_L(L,IND);
```

```
    U_out=U;
 else
    %se l'elemento della matrice W in posizione
    %(is,js) è positivo allora si aggiunge un
    %elemento pari a tau_s nella seconda
    %riga della matrice LU, relativa all'upper
    %bound.
    LU_out(2,js)=tau_s;
    NEG=0;
    POS=1;
    L_out=L;
    U_out=trova_U(U,IND);
 end
```

A.2.6 Trova_L

```
function L_out=trova_L(L,IND)
%trova_L(L,IND) consente di
%valutare il nuovo insieme L.
%*****************************************
%                   SINTASSI
%L è un insieme a cui viene tolto
%un elemento ogni volta che si inserisce un
%valore nella prima riga della matrice LU.
%IND rappresenta l'indice del primo minimo
%trovato nella matrice S.

%si assegna a js il valore dell'indice in
%posizione (1,2) del vettore IND.
js=IND(1,2);
%si assegna a index il valore di L uguale
%a js.
index=find(L==js);
%si assegna a p il numero di righe del
%vettore L e a m il numero di colonne.
[p,m]=size(L);
%se L ha un solo elemento, allora L_out
%conterrà solo 0.
if (m==1)
    L_out=0;
else
```

```
    %se index è uguale a 1, allora L_out
    %sarà uguale a tutto ciò che va
    %dalla posizione 2 fino a m.
    if (index==1)
    L_out=L(2:m);
    %se index=m allora L_out sarà uguale a
    %prima fino all'elemento (m-1).
    else if (index==m)
              L_out=L(1:m-1);
          else
               %per i che varia da 1 a index-1,
               %L_out resta uguale.
               for (i=1:index-1)
               L_out(i)=L(i);
               end
               %per i che varia da index+1 a m,
               %L_out è uguale a prima ma
               %traslato di una posizione in avanti.
               for (i=index+1:m)
               L_out(i-1)=L(i);
               end
            end
    end
end
```

A.2.7 Trova_U

```
function U_out=trova_U(U,IND)
%trova_U(U,IND) consente di valutare
%il nuovo insieme U.
%*********************************************
%                    SINTASSI
%U è un insieme a cui viene tolto un elemento
%ogni volta che si inserisce un valore nella
%seconda riga della matrice LU.
%IND rappresenta l'indice del primo minimo
%della matrice S.

%si assegna a js il valore dell'indice in
%posizione (1,2) del vettore IND.
js=IND(1,2);
%si assegna a index il valore di U uguale a js.
```

```
index=find(U==js);
%si assegna a p il numero di righe del
%vettore U e a m il numero di colonne.
[p,m]=size(U);
%se U ha un solo elemento, allora U_out
%conterrà solo 0.
if (m==1)
    U_out=0;
else
    %se index è uguale a 1, allora U_out sarà
    %uguale a tutto ciò che va
    %dalla posizione 2 fino a m.
    if (index==1)
    U_out=U(2:m);
    %se index=m allora U_out sarà uguale
    %a prima fino all'elemento (m-1).
    else if (index==m)
              U_out=U(1:m-1);
          else
               %per i che varia da 1 a index-1,
               %U_out resta uguale.
               for (i=1:index-1)
               U_out(i)=U(i);
               end
               %per i che varia da index+1 a m,
               %U_out è uguale a prima ma
               %traslato di una posizione in avanti.
               for (i=index+1:m)
               U_out(i-1)=U(i);
               end
            end
    end
end
```

A.2.8 Calcola_W_k

```
function [W_new,k_new]=calcola_W_k(W,k,tau_s,IND,NEG,POS)
%calcola_W_k(W,k,tau_s,IND,NEG,POS)consente di
%determinare la nuova matrice W di n_c righe
%e di m colonne e il nuovo vettore colonna k di n_c
%righe.
%*****************************************************
```

```
%                    SINTASSI
%W è una matrice di n_c righe e m colonne.
%k è un vettore colonna di n_c righe.
%tau_s è l'elemento minimo tra tutti i tau
%valutati al variare dell'indice di riga i.
%IND è la coppia di indici relativa al valore
%minimo della matrice S.
%NEG è una variabile definita a seconda del
%valore dell'elemento(is,js)della matrice W.
%POS è una variabile definita a seconda del
%valore dell'elemento (is,js) della matrice W.

%si assegna a n_c il valore delle righe
%della matrice W e a m il valore delle
%colonne della matrice W.
[n_c,m]=size(W);
%si assegna a is l'elemento (1,1) del
%vettore IND.
is=IND(1,1);
%si assegna a js l'elemento (1,2) del
%vettore IND.
js=IND(1,2);
%si esegue il ciclo for al variare
%dell'indice di riga i e
%dell'indice di colonna j.
for i=1:n_c
   for j=1:m
       if (W(i,js)<0 && NEG==1)
           %se la condizione nell'if è verificata
           %si valuta k attraverso la
           %relazione
           k_new(i,1)=(k(i)+(tau_s*W(i,js)));
           %se l'indice di colonna j per cui
           %si sta valutando il ciclo for è
           %uguale all'indice js allora la
           %matrice W avrà uno zero nella
           %posizione j=js.
            if  (j==js)
                W_new(i,j)=0;
            else
            %se la condizione non è verificata,
            %quindi se l'indice di colonna j è
            %diverso dall'indice js che si sta
```

```
            %considerando allora l'elemento (i,j)
            %di W_new è uguale all'elemento (i,j)
            %della matrice W.
                W_new(i,j)=W(i,j);
            end
         else
        if (W(i,js)>0 && POS==1)
              %se la condizione nell'if è verificata
              %si valuta k attraverso la
              %relazione.
              k_new(i,1)=(k(i)-(tau_s*W(i,js)));
              %se l'indice di colonna j per cui
              %si sta valutando il ciclo for è uguale
              %all'indice js allora la matrice W
              %avrà uno zero nella posizione j=js.
               if (j==js)
                  W_new(i,j)=0;
              else
              %se la condizione non è verificata,
              %quindi se l'indice di colonna j è
              %diverso dall'indice js che si
              %sta considerando allora l'elemento
              %(i,j) della matrice W_new è
              %uguale all'elemento (i,j) della
              %matrice W.
              W_new(i,j)=W(i,j);
         end
              end
              %se non ci si trova in nessuno
              %dei casi precedenti.
        if ((W(i,js)>=0 && POS==0) || (W(i,js)<=0 && NEG==0))
           %se la condizione è verificata il
           %vettore k e la matrice W
           %saranno uguali a quelli precedenti,
           %ovvero quelli presenti in ingresso.
            k_new(i,1)=k(i);
            W_new(i,j)=W(i,j);
         end
    end
end
end
```

A.2.9

A.3 Algoritmo HMB

```
algoritmo_HB;
L0=L_vett;
U0=U_vett;
mu=trova_mu(vettore_tau);
X=trova_X(A,LU);

for (s=1:(mu-1))
js=vettore_js(1,s);
INT_MIN=trova_INT_MIN(W,k,js,X);
POS=trova_POS(LU,js,vettore_tau,s);
LU_max=trova_LU_max(W,LU,INT_MIN,js,vettore_tau,POS,s);
LU=LU_max;
end

L_max=LU(1,:)
U_max=LU(2,:)
```

A.3.1 Nuovo script per l'Algoritmo HB

```
clear all
W_in=[];
k_in=[];
X_opt=[];
[k,W]=data_input(k_in,W_in,X_opt);
A=W;
b=k;
W_in=W;
k_in=k;
[n_c,m]=size(W);
LU=zeros(2,m);
L=1:m;
U=1:m;
vettore_tau=zeros(1,2*m);
vettore_js=zeros(1,2*m);

for (s=1:2*m)
```

```
tau_s=calcola_tau_s(A,b);
S=calcola_S(W,k,tau_s);
IND=calcola_IND(S);
[LU_out,NEG,POS,L_out,U_out]=calcola_LU(W,tau_s,IND,LU,L,U);
[W_new,k_new]=calcola_W_k(W,k,tau_s,IND,NEG,POS);
LU=LU_out
L=L_out;
U=U_out;
W=W_new;
k=k_new;
A=W;
b=k;
vettore_tau(1,s)=tau_s
vettore_js(1,s)=IND(1,2)
end

L_vett=LU(1,:)
U_vett=LU(2,:)
W=W_in;
A=W_in;
k=k_in;
```

A.3.2 Trova_mu

```
function mu=trova_mu(vettore_tau)
%trova_mu(vettore_tau) consente di
%determinare l'indice in cui si trova il
%valore massimo del vettore vettore_tau
%in cui vi sono tutti i valori di tau_s
%trovati per mezzo della funzione calcola_tau_s.
%************************************************
%                  SINTASSI
%vettore_tau è il vettore contenente
%tutti i tau_s.

%si assegna a MASSIMO_TAU il valore massimo
%del vettore vettore_tau.
MASSIMO_TAU=max(vettore_tau);
%si ricercano all'interno di vettore_tau
%tutti gli elementi uguali a
%MASSIMO_TAU. INDICI fornisce gli indici
%degli elementi massimi di vettore_tau.
```

```
INDICI_MAX=find(vettore_tau==MASSIMO_TAU);
%si assegna a mu il primo indice trovato.
mu=INDICI_MAX(1,1);
```

A.3.3 Trova_X

```
function X=trova_X(A,LU)
%trova_X(A,LU) consente di determinare
%la matrice X. Ogni colonna di tale matrice
%costituisce un vettore n_c dimensionale
%detto vettore estremo di C(A,b).
%***********************************************
%                  SINTASSI
%A è una matrice di n_c righe e m colonne.
%LU è la matrice che contiene nella prima
%riga tutti i termini che definiscono il
%lower bound e nella seconda riga contiene
%i termini che definiscono l'upper bound.

%si assegna a n_c il numero di righe della
%matrice A e a m il numero di
%colonne della matrice A.
[n_c,m]=size(A);
for(i=1:n_c)
    for(j=1:m)
        %se l'elemento della matrice A in
        %posizione (i,j) è maggiore di 0
        %allora nella posizione (i,j) della
        %matrice X si avrà un termine uguale
        %all'elemento presente nella seconda
        %riga della matrice LU nella colonna j.
    if(A(i,j)>0);
        X(i,j)=LU(2,j);
    else
        %se l'elemento di A in posizione (i,j)
        %è minore di 0,la matrice X avrà nella
        %posizione (i,j) un termine uguale
        %all'elemento presente nella prima
        %riga della matrice LU nella colonna j.
        if(A(i,j)<=0)
            X(i,j)=LU(1,j);
        end
```

```
    end
end
end
```

A.3.4 Trova_INT_MIN

```
function INT_MIN=trova_INT_MIN(W,k,js,X)
%trova_INT_MIN(W,k,js,X) consente di
%valutare il minimo valore intero tra
%tutti quelli trovati al variare degli
%elementi di riga ma per la stessa colonna
%della matrice W.
%*********************************************
%                    SINTASSI
%W è una matrice di n_c righe e m colonne.
%k è un vettore colonna di n_c righe.
%js è un indice di riga scelto tra gli
%elementi del vettore vettore_js.
%X è una matrice di n_c righe e m colonne.

%si assegna a n_c il numero di righe di W e
%a m il numero di colonne della matrice W.
[n_c,m]=size(W);
for (i=1:n_c)
%si assegna a w_ijs il valore assoluto
%dell'elemento che si trova alla riga i e
%alla colonna js della matrice W.
w_ijs=abs(W(i,js));
%INTERO è un vettore colonna e in ogni riga
%si ha l'elemento calcolato
%con la formula allo step i-esimo.
INTERO(i,:)=floor((k(i)-W(i,:)*(X(i,:))')/w_ijs);
end
%tra tutti i valori di INTERO si sceglie
%quello minore e si assegna a
%INT_MIN tale valore.
INT_MIN=min(INTERO);
```

A.3.5 Trova_POS

```
function POS=trova_POS(LU,js,vettore_tau,s)
%trova_POS(LU,js,vettore_tau,s) consente di
%definire la variabile POS.
%*********************************************
%                      SINTASSI
%LU è la matrice contenente nella prima
%riga il lower bound e nella seconda riga
%l'upper bound.
%js è un indice di colonna.
%vettore_tau è il vettore contenente tutti
%i tau_s valutati per mezzo della
%funzione calcola_tau_s.
%s indica lo step che si sta eseguendo.

%se l'elemento nella seconda riga della
%matrice LU in posizione js
%(l'upper bound nella colonna js) è uguale
%all'elemento di vettore_tau
%nella stessa colonna dello step
%considerato, POS=1.
if(LU(2,js)==vettore_tau(1,s))
    POS=1;
else
    %se LU(2,js) è diverso dall'elemento
    %di vettore_tau nella colonna s,POS=0.
    POS=0;
end
```

A.3.6 Boundaries massimali

```
function LU_max=
trova_LU_max(W,LU,INT_MIN,js,vettore_tau,POS,s)
%trova_LU_max(W,LU,INT_MIN,js,vettore_tau,POS,s)
%determina la matrice LU_max che contiene
%nella prima riga il vettore che definisce
%il lower bound e nella seconda riga il
%vettore che definisce l'upper bound.
%*************************************************
%                       SINTASSI
```

```
%W è una matrice di n_c righe e m colonne.
%LU è la matrice che contiene nella prima
%riga il lower bound e nella seconda riga
%l'upper bound.
%INT_MIN è il minimo valore intero trovato
%con la funzione trova_INT_MIN.
%js è un indice scelto nel vettore
%vettore_js nella colonna indicata dallo
%step s che si sta eseguendo.
%vettore_tau è il vettore che contiene
%tutti i tau_s trovati con il file
%algoritmo_HB.
%POS è una variabile definita in base
%al valore della seconda riga della
%matrice LU alla colonna js.
%s indica lo step che si sta eseguendo.

%si assegna a n_c il numero di righe della
%matrice W e a m il numero di
%colonne della matrice W.
[n_c,m]=size(W);
LU_max=LU;
for j=1:m
    %se l'indice di colonna j è diverso
    %dall'indice js allora gli elementi della
    %matrice LU restanno uguali per quei
    %valori di j diversi da js.
       if(j~=js)
          LU_max(2,j)=LU(2,j);
          LU_max(1,j)=LU(1,j);
       end
       %se l'indice j è uguale all'indice js,
       %allora si valuterà l'elemento da
       %posizionare nella colonna js della
       %matrice LU, nella seconda riga se POS=1,
       %nella prima riga se POS=0.
       if(j==js)
            if(POS==1)
                %se il valore di tau_s
                %presente nella colonna s del
                %vettore vettore_tau è minore
                %del valore di tau_s presente
                %nella colonna (s+1) di
```

```
                %vettore_tau, il valore della
                %matrice LU_max è uguale al
                %valore presente precedentemente.
               if(vettore_tau(1,s)<vettore_tau(1,(s+1)))
                  LU_max(2,js)=LU(2,js);
               end
               %se il valore di tau_s nella
               %colonna s è uguale al valore di
               %tau_s nella colonna (s+1),
               %si somma al valore precedente di
               %LU nella colonna js nella
               %seconda riga, il valore INT_MIN.
               if(vettore_tau(1,s)==vettore_tau(1,(s+1)))
                  LU_max(2,js)=LU(2,js)+INT_MIN;
               end
            end

            if(POS==0)
                %se il valore di vettore_tau in
                %colonna s è minore del valore
                %alla colonna (s+1),
                %l'elemento nella colonna js
                %della prima riga è uguale
                %al valore assunto precedentemente.
               if(vettore_tau(1,s)<vettore_tau(1,(s+1)))
                  LU_max(1,js)=LU(1,js);
               end
               %se il valore di vettore_tau
               %nella colonna s e nella
               %colonna (s+1) sono uguali,
               %si sottrae al valore della prima
               %riga nella colonna js, il valore INT_MIN.
               if(vettore_tau(1,s)==vettore_tau(1,(s+1)))
                  LU_max(1,js)=LU(1,js)-INT_MIN;
               end
            end
      end
end
```

A.4 Script del caso non singleton

```
algoritmo_HB
algoritmo_HMB
W_in=[];
P=[];
[W_out,k_out]=trova_W_out_k_out(W_in,P,X)
LU_def=trasla_LU(LU,X_opt)
```

A.4.1 Ricerca delle GMEC decentralizzate

```
function [W_out,k_out]=trova_W_out_k_out(W_in,P,X)
%trova_W_out_k_out(W_in,P,X) prende in ingresso
%la matrice W_in delle GMEC, la matrice P
%della suddivisione dei posti e la matrice X e
%determina la matrice W_out e il vettore k_out
%che consentono di definire un nuovo insieme di GMEC.
%************************************************
%                  SINTASSI
%W_in è una matrice di dimensione q righe
%e m colonne, costituita dai coefficienti delle GMEC.
%P è una matrice che definisce la suddivisione
%dei posti nel caso in cui non siano singleton.
%Ogni riga della matrice P corrisponde a un insieme
%di posti e ogni colonna corrisponde a un posto,
%dal posto p1 al posto pm. Si avrà 1 dove
%il posto è presente nell'insieme e
%0 se il posto non è presente.
%X è una matrice di n_c righe e m colonne.

%si assegna a q il numero di righe della matrice
%W_in e a m il numero di colonne della matrice W_in.
[q,m]=size(W_in);
%si assegna a nu il numero di righe della matrice P
%e a m il numero di colonne della matrice P.
[nu,m]=size(P);
[n_c,m]=size(X);
%per r che varia da 1 al numero di righe
%della matrice W_in.
for r=1:q
%per i che varia lungo le righe della matrice P.
```

```
for i=1:nu
%per j che varia lungo le colonne della matrice P.
for j=1:m
%se l'elemento (i,j) della matrice P vale 1,
%cioè l'insieme i contiene il posto j.
if   ((P(i,j)==1))
    %la matrice W_out in uscita avrà
    %alla riga (i+(r-1)*nu) e alla colonna
    %j, l'elemento della riga r e della colonna j
    %della matrice W_in.
 W_out(i+(r-1)*nu,j)=W_in(r,j);
else
    %se l'elemento in P non è presente, in uscita W_out
    %in posizione (i+(r-1)*nu,j) avrà 0.
    W_out(i+(r-1)*nu,j)=0;
end
end
end
end
%si considera una nuova matrice X_new di
%dimensione uguale a W_in in cui vi sono le
%stesse colonne di prima ma solo le prime q righe.
X_new=X((1:q),:);
%per r che varia da 1 al numero di righe
%della matrice X_new.
for r=1:q
    %per i che varia da 1 a nu.
    for i=1:nu
        %il vettore k_out in uscita sarà costituito
        %dagli elementi trovati con la seguente
        %espressione.
        k_out(i+(r-1)*nu)=W_out(i+(r-1)*nu,:)*X_new(r,:)';
        end
end
%poichè k_out è un vettore riga si esegue
%una traslazione di tale vettore.
k_out=k_out';
```

A.4.2 Boundaries definitivi

```
function LU_def=trasla_LU(LU,X_opt)
%trasla_LU(LU,X_opt) determina la matrice
```

```
%LU_def dei boundaries nell'originale
%sistema di riferimento.
%*******************************************
%                       SINTASSI
%LU è la matrice dei boundaries ottenuta
%dopo aver applicato l'algoritmo per la
%ricerca dei boundaries massimali.
%X_opt è un vettore colonna di (m+1) componenti
%determinato risolvendo un problema di
%programmazione lineare intero; le prime m
%componenti rappresentano le coordinate del
%centro di un sistema di riferimento
%traslato rispetto a quello fornito dalla GMEC.

[d,m]=size(LU);
[t,u]=size(X_opt);
m=(t-1);
LU_def=zeros(2,m);
%si definisce C un vettore colonna costituito
%da m componenti uguali alle prime m
%componenti del vettore X_opt.
C=X_opt(1:m,1);
 %al variare dell'indice di colonna j da 1 a m.
         for j=1:m
             %la matrice LU dei boundaries per
             %il sistema originale si ottiene
             %sommando ai bound di ciascun
             %posto il valore di cui è stato
             %traslato.
        LU_def(:,j)=LU(:,j)+C(j);
         end
```

A.5 Conteggio delle marcature

A.5.1 Definizione della matrice LU con soli elementi positivi

```
function LU=limita_bound(LU)
%limita_bound(LU) determina una nuova matrice
%dei boundaries con soli elementi maggiori
%o uguali a zero affinchè possa racchiudere
```

```
%solo marcature positive.
%************************************************
%                     SINTASSI
%LU è la matrice dei boundaries determinati
%con l'algoritmo.

[d,m]=size(LU);
%per l'indice di colonna j che varia da 1 a m.
for j=1:m
%per l'indice di riga i che varia da 1 a d.
    for i=1:d
        %se l'elemento (i,j) della matrice
        %LU è minore di 0.
if(LU(i,j)<0)
    %si fa assumere all'elemento (i,j)
    %della matrice LU il valore 0, valore minimo
    %possibile per ottenere marcature positive.
    LU(i,j)=0;
end
end
end
```

A.5.2 Punti di intersezione delle GMEC con gli assi dei posti

```
function index=trova_index(W_in,k_in)
%trova_index(W_in,k_in) determina il valore
%minimo e il valore massimo dell'
%intersezione delle GMEC con ciascun posto.
%*******************************************
%                     SINTASSSI
%W_in è una matrice con tante righe quante
%sono le GMEC singole e tante colonne
%quanti sono i posti.
%k_in è un vettore colonna con tante
%righe quante sono le GMEC singole.

[q,m]=size(W_in);
%per j che varia lungo le colonne della
%matrice W_in.
for j=1:m
    %per i che varia lungo le righe della
    %matrice W_in.
```

```
    for i=1:q
        %si esegue il rapporto tra l'elemento
        %i del vettore k_in e l'elemento
        %(i,j) della matrice W_in e si
        %considera il valore intero.
       M(i,j)=floor(k_in(i)/W_in(i,j));
    end
end
%si definisce index_min il vettore costituito
%dai valori minimi di ogni colonna.
    index_min=min(M);
    %si definisce index_max il vettore
    %costituito dagli elementi massimi di
    %ciascuna colonna.
    index_max=max(M);
%la matrice index è costituita dalla
%concatenazione verticale del vettore
%index_min che contiene gli elementi minimi
%e del vettore index_max che contiene gli
%elementi massimi.
    index=[index_min;index_max];
```

A.5.3 Esempio di script per una rete 3 posti

```
clear all
non_singleton;
LU=LU_def;
W_in=[];
k_in=[];
LU=limita_bound(LU);
[q,m]=size(W_in);
marcature=0;
 for x1=LU(1,1):LU(2,1)
    for x2=LU(1,2):LU(2,2)
        for x3=LU(1,3):LU(2,3)
            X=[x1;x2;x3];
            Valid=zeros(1,q);
            for i=1:q
                if ( W_in(i,:)*X <= k_in(i))
                    Valid(i)=1;
                else
                    Valid(i)=0;
```

```
                end
            end
            if (Valid==ones(1,q))
                marcature=marcature+1;
            end
        end
    end
 end
marcature
```

Bibliografia

[1] F. Basile, A. Giua, and C. Seatzu. Decentralized supervisory control of Petri nets based on monitor places. In preparation, 2009.

[2] A. Di Febbraro and A. Giua. *Sistemi ad eventi discreti*. McGraw-Hill, 2002.

[3] A. Bemporad, C. Filippi, and F.D. Torrisi. Inner and outer approximation of polytopes using boxes. *Computational Geometry*, 27:151-178,2004.

[4] M.S. Bazaraa and J.J. Jarvis. *Linear programming and network flows*. Wiley, 1994.

[5] M.V. Iordache and P.J. Antsaklis. Decentralized control of Petri nets with constraint transformations. *IEEE Transaction on Automatic Control*, 51(2):376-381, February 2006.

[6] M.V. Iordache and P.J. Antsaklis. Admissible decentralized control of Petri nets. *Proc. 2003 American Control Conference (Denver,Colorado)*, pages 332-337, June 2003.

[7] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transaction on Automatic Control*, 37(11):1692-1708, November 1992.

[8] A. Giua, F. DiCesare and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. *1992 IEEE Internation Conference on System, Man and Cybernetics (Chicago,Illinois)*, pages 974-979, October 1992.

[9] F. Basile, A. Giua, and C. Seatzu. Some new results on supervisory control of Petri nets with decentralized monitor places. 17^{th} *IFAC World Congress, Seoul, Korea*, July 2008.

[10] F. Basile, A. Giua, and C. Seatzu. Supervisory control of Petri nets with decentralized monitor places. 26^{th} *American Control Conference (ACC'07), New York, USA*, pages 4657-4962, July 2007.

[11] F. Basile, A. Giua, and C. Seatzu. Decentralized supervisory control of Petri nets with monitor places. 10^{th} *IEEE International Conference on Emerging Technologies and Factory Automatic (ETFA'05), Catania, Italy*, Settembre 2005.

[12] G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Transaction on Automatic Control*, 45(9):1620-1638, 2000.

[13] K. Rudie, S. Lafortune and F. Lin. Minimal communication in a distributed discrete-event system. *IEEE Transaction on Automatic Control*, 48(6):957-975, June 2003.

[14] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transaction on Automatic Control*, 35(12):1330-1337, December 1990.

[15] J.O. Moody and P.J. Antsaklis. Petri net supervisors for DES with uncontrollable and unobservable transitions. *IEEE Transaction on Automatic Control*, 45(3):462-476, March 2000.