Cammino minimo per agenti multipli

Università degli Studi di Parma

Nome: Giorgia Tedaldi
Matricola: 339642
Corso: Ricerca Operativa

1 Introduzione

L'obiettivo di questo progetto è quello di trovare una soluzione al seguente problema. Sia dato un grafo non orientato con archi aventi tutti tempo di percorrenza pari a 1. Su questo grafo si muovono n agenti. Ogni agente i ha nodo di origine oi e un nodo destinazione di. Si definisca un modello matematico per minimizzare la somma dei cammini minimi di tutti gli agenti, tenendo conto che gli agenti non possono trovarsi nello stesso istante in un nodo o lungo un arco.Si formuli il modello matematico in AMPL e si definiscano i dati di una particolare istanza, risolvendola. Si faccia inoltre un'analisi di cosa succede se si modificano alcuni dei dati dell'istanza.

L'idea generale dell'approccio utilizzato si basa sull'utilizzo di variabili binarie e a valori interi. Quelle del primo tipo sono utilizzate per tenere traccia del percorso effettuato da un certo agente a; queste sono settate a 1 per ogni nodo v e ogni arco e che l'agente utilizza per arrivare a destinazione. Le variabili a valori interi invece servono nella gestione delle collisioni. Ognuna indica l'istante temporale in cui l'agente a entra ed esce da un certo nodo v. In particolare si assume che ogni agente a si sposti ad ogni istante di tempo (non può fermarsi ad aspettare su un nodo) e che una volta giunto a destinazione esca dal grafo, sono quindi ammesse destinazioni uguali per agenti diversi, (purchè entrino in istanti temporali diversi). L'obiettivo del problema è quello di minimizzare la somma dei cammini minimi; s'intende cioè cercare, per ogni agente, il cammino minimo per andare dalla sorgente alla destinazione, tale da evitare collisioni con gli altri agenti.

Un secondo possibile approccio potrebbe essere quello di fare uso di sole variabili binarie: definito un insieme di nodi, un insieme di archi e un insieme di istanti temporali, (da 0 a m, dove m è la cardinalità dell'insieme dei nodi), la variabile indica se l'agente a occupa il nodo v all'istante t. La difficoltà nella realizzazione di questo approccio risiede nella gestione contemporanea delle 3 dimensioni.

2 Modello Matematico

Definiamo:

- $VERTEX$ = insieme dei *nodi*;
- $EDGE$ = insieme degli *archi*;
- $AGENT$ = insieme degli *agenti*;
- $\forall a \in AGENT$ definiamo $Source(a) \in VERTEX$ come il nodo di partenza;
- $\forall a \in AGENT$ definiamo $Destination(a) \in VERTEX$ come il nodo di arrivo;
- $\forall a \in AGENT, \forall (i,j) \in EDGE$ introduciamo una variabile booleana $used_{(i,j)a}$ che indica se l'arco *(i,j)* è utilizzato dall'agente a;
- $\forall a \in AGENT, \forall v \in VERTEX$ la variabile booleana $flow_{va}$ indica se l'agente a passa attraverso il nodo v;
- $\forall v \in VERTEX, \forall a \in AGENT$ introduciamo una variabile a valori interi in_time_{va} che indica in quale istante temporale l'agente a entra nel nodo v;
- $\forall v \in VERTEX, \forall a \in AGENT$ introduciamo una variabile a valori interi out_time_{va} che indica in quale istante temporale l'agente a esce dal nodo v;

Tutte le equazioni necessarie alla costruzione del modello (vincoli per la conservazione del flusso, per la descrizione delle relazioni temporali tra gli istanti di entrata/uscita da un nodo e per la gestione delle collisioni) sono le seguenti:

$$\forall a \in AGENT : flow_{Source(a)a} = 1 \tag{1}$$

$$\forall a \in AGENT : flow_{Destination(a)a} = 1 \tag{2}$$

$$\forall a \in AGENT,\ \forall i \in VERTEX \mid (i, Source(a)) \in EDGE : used_{(i,Source(a))a} = 0 \tag{3}$$

$$\forall a \in AGENT,\ \forall i \in VERTEX \mid (Destination(a), i) \in EDGE : used_{(Destination(a),i)a} = 0 \tag{4}$$

$$\forall a \in AGENT, \forall i \in VERTEX \setminus \{Source(a)\} : \sum_{j \in VERTEX\,:\,(j,i) \in EDGE} used_{(j,i)a} = flow_{ia} \tag{5}$$

$$\forall a \in AGENT, \forall i \in VERTEX \setminus \{Destination(a)\} : \sum_{j \in VERTEX\,:\,(i,j) \in EDGE} used_{(i,j)a} = flow_{ia} \tag{6}$$

$$\forall a \in AGENT : in\_time_{Source(a)a} = 1 \tag{7}$$

$$\forall a \in AGENT, \forall i \in VERTEX : out\_time_{ia} = in\_time_{ia} + flow_{ia} \tag{8}$$

$$\forall (i,j) \in EDGE, \forall a \in AGENT : used_{(i,j)a} = 1 \Rightarrow out\_time_{ia} = in\_time_{ja} \tag{9}$$

$$\forall a_1, a_2 \in AGENT \mid a_1 \neq a_2, \forall i \in VERTEX : flow_{ia_1} = 1 \wedge flow_{ia_2} = 1 \Rightarrow in\_time_{ia_1} \neq in\_time_{ia_2} \tag{10}$$

$$\forall a_1, a_2 \in AGENT \mid a_1 \neq a_2, \forall (i,j) \in EDGE : used_{(j,i)a_1} = 1 \wedge used_{(i,j)a_2} = 1 \Rightarrow out\_time_{ja_1} \neq out\_time_{ia_2} \tag{11}$$

$$\forall a \in AGENT, \forall i,j \in VERTEX \mid i \neq j : in\_time_{ja} \neq in\_time_{ia} \tag{12}$$

$$\forall a \in AGENT, \forall i,j \in VERTEX \mid i \neq j : out\_time_{ja} \neq out\_time_{ia} \tag{13}$$

I vincoli **(1)** e **(2)** specificano che i nodi sorgente/destinazione sono occupati dal rispettivo agente.
I vincoli **(3)** e **(4)** impongono ad ogni agente di **uscire** (rispettivamente **entrare**) dalla sua sorgente (nella sua destinazione): tutti gli archi entranti in *Source(a)* (uscenti da *Destination(a)*) non sono infatti utilizzati da *a*.

Se per completare il suo percorso l'agente a attraversa altri nodi, allora deve **entrare e uscire** esattamente una volta da ognuno di essi. Questo comportamento viene descritto dai vincoli **(5)** e **(6)**.

Imponiamo con il vincolo **(7)** che l'istante temporale iniziale (il tempo in cui ogni agente entra nella rispettiva sorgente) sia 1.

Per descrivere la relazione che esiste tra tempi di entrata/uscita da un nodo si utilizza il vincolo **(8)**: se l'agente a occupa il nodo i allora l'instante in cui a esce da i è pari all'istante in cui vi entra aumentato di un'unità.

Per descrivere, invece, il tempo di attraversamento di un arco si utilizza il vincolo **(9)**: se l'agente a passa attraverso l'arco *(i,j)* l'istante in cui avviene tale azione è pari al tempo in cui a entra nel nodo *j*. Questo vale poichè tutti gli archi hanno lo stesso costo.

Per quanto riguarda la gestione delle collisioni si distinguono due casi:

- **conflitto di nodo**: due agenti non possono trovarsi sullo stesso nodo nello stesso momento;
- **conflitto di arco**: due agenti non possono trovarsi sullo stesso arco nello stesso momento, ovvero non possono scambiarsi di posizione.

Il vincolo **(10)** specifica che: considerando due qualsiasi agenti a_1 e a_2, se questi occupano uno stesso nodo i durante il loro cammino, allora gli istanti in cui entrano in i devono essere diversi.

Avendo vincolato la presenza di **al massimo** un agente in uno specifico nodo allora due agenti non potranno mai attraversare lo stesso arco nella stessa direzione. Al contrario, supponiamo che un qualsiasi arco *(i,j)* sia attraversato in direzioni opposte da due diversi agenti a_1 e a_2, allora i tempi di uscita dai rispettivi nodi di partenza devono essere diversi, come specificato nel vincolo **(11)**.

Per completare il modello, attraverso i vincoli **(12)** e **(13)**, si specifica che ogni agente può **entrare** in (rispettivamente **uscire da**) **un solo** nodo per ogni instante di tempo.

3 Modello AMPL

In questo paragrafo viene spiegato come il modello matematico è stato tradotto nel linguaggio AMPL. Prima di procedere è utile un chiarimento su come è stata linearizzata la **disuguaglianza stretta** tra variabili.

3.1 Linearizzazione della disuguaglianza stretta tra due variabili

Per ogni vincolo in cui vi è la necessità di imporre una disguaglianza stretta (in questo caso la differenza) tra due variabili x e y si introduce una variabile **booleana** δ di supporto e si sfrutta un coefficiente M, scelto sufficientemente grande. Essendo l'obiettivo del problema quello di minimizzare la somma dei cammini minimi di ogni agente si deduce che un qualsiasi agente potrà **al massimo** attraversare una volta tutti i nodi; per questo motivo possiamo definire $M = k + 1$, dove k è la cardinalità dell'insieme dei nodi.
La disuguaglianza stretta si esprimerà, quindi, attraverso due vincoli:

$$x <= y - 1 + M * \delta$$

$$x >= y + 1 - M(1 - \delta)$$

L'unico modo per cui entrambi i vincoli risultano soddisfatti è se vale una delle seguenti condizioni $x < y$ o $x > y$. Supponiamo che $x = y$:

- il primo vincolo diventa $0 <= -1 + M * \delta$, per cui necessariamente $\delta = 1$ se vogliamo che il vincolo sia valido. Sostituendo M otteniamo $0 <= k$, che è sempre verificato;
- il secondo vincolo, di conseguenza, diventa $0 >= 1$ che è falso.

Quindi affinchè i vincoli siano validi deve necessariamente esistere un rapporto di stretta disuguaglianza tra le due variabili.

3.2 Traduzione dei vincoli in AMPL

I vincoli **(1)**, **(2)**, **(3)**, **(4)**, **(5)**, **(6)**, **(7)** e **(8)**, descritti nel *paragrafo 2*, non necessitano di particolari modifiche o aggiunte e si traducono nel linguaggio AMPL nel seguente modo:

```
subject to source_constraint_1{a in AGENT}:
    flow[Source[a], a] = 1;
```

Listing A: Traduzione vincolo **(1)**

```
subject to destination_constraint_1{a in AGENT}:
    flow[Destination[a], a] = 1;
```

Listing B: Traduzione vincolo **(2)**

```
subject to source_constraint_2{i in VERTEX, a in AGENT: (i,Source[a]) in EDGE}:
    used[i, Source[a], a] = 0;
```

Listing C: Traduzione vincolo **(3)**

```
subject to destination_constraint_2{i in VERTEX, a in AGENT: (Destination[a], i) in EDGE}:
    used[Destination[a], i, a] = 0;
```

Listing D: Traduzione vincolo **(4)**

```
subject to enter_vertex_contraint{i in VERTEX, a in AGENT: i != Source[a]}:
    sum{j in VERTEX: (j,i) in EDGE}used[j, i, a] = flow[i, a];
```

Listing E: Traduzione vincolo **(5)**

```
subject to leave_vertex_constraint{i in VERTEX, a in AGENT: i != Destination[a]}:
    sum{j in VERTEX: (i,j) in EDGE}used[i, j, a] = flow[i, a];
```

Listing F: Traduzione vincolo **(6)**

```
subject to source_in_time{a in AGENT}:
    in_time[Source[a],a] = 1;
```

Listing G: Traduzione vincolo **(7)**

```
subject to out_time_constraint{i in VERTEX, a in AGENT}:
    out_time[i, a] = in_time[i, a] + flow[i,a];
```

Listing H: Traduzione vincolo **(8)**

I vincoli successivi, invece, presentano operatori logici e operatori come la disuglianza stretta, che devono essere manipolati e linearizzati affinchè possano essere tradotti in AMPL.

Il vincolo **(9)**, la cui traduzione è rappresentata in *Listing I* , presenta l'implicazione logica *se.. allora* seguita da un'uguaglianza tra variabili. Il vincolo viene spezzato in due differenti equazioni, una di <= e una di >=, in modo che insieme impongano la sola uguaglianza. Tale uguaglianza deve valere quando la variabile *used* è pari a 1, cioè quando l'arco considerato è usato dall'agente durante il suo cammino; è necessario costruire le due equazioni in modo che quando $used = 0$ i vincoli risultino non significativi per il modello.

Il vincolo *edge_time_constraint_1* nel momento in cui $used = 0$ diventa $in\_time[j, a] <= M$, vincolo che non influisce sul modello, infatti il massimo *in_time* possibile è pari alla cardinalità dell'insieme dei nodi, quindi sicuramente minore di M, proprio per come è stata definita tale costante (vedi paragrafo 3.1).

Allo stesso modo il vincolo *edge_time_constraint_2* diventa superfluo quando $used = 0$, poichè $in\_time[j, a] >= 0$. Nel caso in cui, invece, $used = 1$ otteniamo esattamente il vincolo desiderato.

```
subject to edge_time_constraint_1{j in VERTEX, i in VERTEX, a in AGENT: (i, j) in EDGE}:
    in_time[j,a] <= used[i,j,a]*(out_time[i, a]) + M*(1 - used[i, j, a]);

subject to edge_time_constraint_2{j in VERTEX, i in VERTEX, a in AGENT: (i, j) in EDGE}:
    in_time[j,a] >= used[i,j,a]*(out_time[i, a]);
```

Listing I: Traduzione vincolo **(9)**

Per il vincolo **(10)** (traduzione in *Listing J*), invece, abbiamo una condizione logica del tipo *se.. e contemporaneamente.. allora* seguita da una disuguaglianza stretta tra variabili. Seguendo la procedura per la linearizzazione della disuguaglianza stretta (descritta nel paragrafo 3.1), si introduce la variabile di supporto *different_nodes* e si spezza il vincolo in due diverse equazioni, descritte sotto. Le equazioni 'differiscono' leggermente dalla procedura standard poichè è necessario imporre che tale disuguaglianza valga sotto la condizione $flow_{ia_1} = 1 \wedge flow_{ia_2} = 1$. In questo modo, se la condizione non è verificata, i vincoli diventano superflui:

- se $flow_{ia_1} = 0 \wedge flow_{ia_2} = 0$ allora *vertex_constraint_1* diventa $0 <= M$, mentre *vertex_constraint_2* diventa $0 >= 0$;
- se $flow_{ia_1} = 1 \wedge flow_{ia_2} = 0$ allora *vertex_constraint_1* diventa $in\_time[i, a1] <= M$, mentre *vertex_constraint_2* diventa $in\_time[i, a1] >= 0$;
- se $flow_{ia_1} = 0 \wedge flow_{ia_2} = 1$ allora *vertex_constraint_1* diventa $0 <= in\_time[i, a2] - 1 + M$, mentre *vertex_constraint_2* diventa $0 >= in\_time[i, a2] + 1 - M$;
- se $flow_{ia_1} = 0 \wedge flow_{ia_2} = 1$ allora *vertex_constraint_1* diventa $in\_time[i, a1] <= in\_time[i, a2] - 1$, mentre *vertex_constraint_2* diventa $in\_time[i, a1] >= in\_time[i, a2] + 1$, che ci impone esattamente la disguaglianza stretta cercata;

```
subject to vertex_collision_1{i in VERTEX, a1 in AGENT, a2 in AGENT: a1 < a2}:
    flow[i, a1] * in_time[i, a1] <= flow[i, a2] * (in_time[i, a2] - 1 +
↪  M*different_nodes[i, a1, a2]) + (1 - flow[i, a2])*M;

subject to vertex_collision_2{i in VERTEX, a1 in AGENT, a2 in AGENT: a1 < a2}:
    flow[i, a1] * in_time[i, a1] >= flow[i, a2] * (in_time[i, a2] + 1 - M*(1 -
↪  different_nodes[i, a1, a2]));
```

Listing J: Traduzione vincolo **(10)**

Si utilizza la tecnica appena descritta anche per il vincolo **(11)** (tradotto in *Listing K*) con le seguenti differenze: al posto della variabile *flow* sarà usata la variabile *used*, al posto della variabile *in_time* la variabile out_time e la variabile d'ausilio si chiamerà *different_edges*.

```
subject to edge_collision_1{i in VERTEX, j in VERTEX, a1 in AGENT, a2 in AGENT: a1 < a2 and
↪  (i, j) in EDGE}:
    used[j, i, a1] * out_time[j, a1] <= used[i, j, a2] * (out_time[i, a2] - 1 +
↪  M*different_edges[i, j, a1, a2]) + (1 - used[i, j, a2])*M;

subject to edge_collision_2{i in VERTEX, j in VERTEX, a1 in AGENT, a2 in AGENT: a1 < a2 and
↪  (i, j) in EDGE}:
    used[j, i, a1] * out_time[j, a1] >= used[i, j, a2] * (out_time[i, a2] + 1 - M*(1 -
↪  different_edges[i, j, a1, a2]));
```

Listing K: Traduzione vincolo **(11)**

I vincoli **(12)** e **(13)** (tradotti rispettivamente in *Listing L* e *Listing M* sono disuguaglianze strette tra variabili, quindi implentate come descritto nel paragrafo 3.1

```
subject to no_multiple_out_time1{i in VERTEX, j in VERTEX, a in AGENT: i != j}:
    out_time[j, a] <= out_time[i, a] - 1 + M*different_out_time[i, j, a];

subject to no_multiple_out_time2{i in VERTEX, j in VERTEX, a in AGENT: i != j}:
    out_time[j, a] >= out_time[i, a] + 1 - M*(1 - different_out_time[i, j, a]);
```

Listing L: Traduzione vincolo **(12)**

```
subject to no_multiple_in_time_1{i in VERTEX, j in VERTEX, a in AGENT: i != j}:
    in_time[j, a] <= in_time[i, a] - 1 + M*different_in_time[i, j, a];

subject to no_multiple_in_time_2{i in VERTEX, j in VERTEX, a in AGENT: i != j}:
    in_time[j, a] >= in_time[i, a] + 1 - M*(1 - different_in_time[i, j, a]);
```

Listing M: Traduzione vincolo **(13)**

4 Conclusioni

4.1 Risoluzione di un'istanza di piccole dimenisoni

Per chiarire il funzionamento del modello costruiamo un esempio come di seguito:

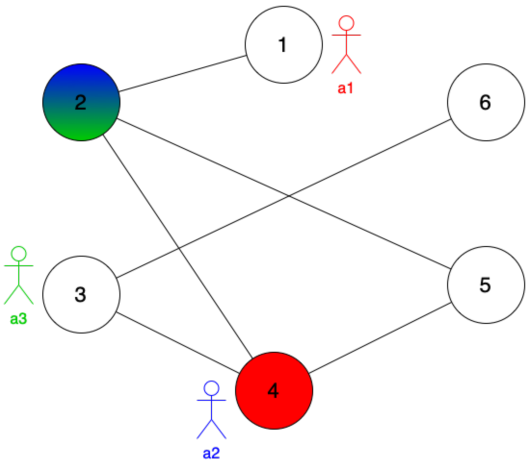

Figure 1: Grafo di esempio.

- sia $VERTEX = \{1, 2, 3, 4, 5, 6\}$ l'insieme dei *nodi*;
- sia $EDGE = \{(1, 2), (2, 1), (2, 4), (4, 2), (2, 5), (5, 2), (3, 6), (6, 3), (3, 4), (4, 3), (4, 5), (5, 4)\}$ l'insieme degli *archi*;
- sia $AGENT = \{1, 2, 3\}$ l'insieme degli *agenti* (indicati in figura con *a1, a2, a3* per maggiore chiarezza) che si muovono sul grafo;
- La sorgente dell'agente *a1* è il nodo *1*, la destinazione il nodo *4*;
- La sorgente dell'agente *a2* è il nodo *4*, la destinazione il nodo *2*;
- La sorgente dell'agente *a3* è il nodo *3*, la destinazione il nodo *2*;

Risolvendo l'istanza si ottengono i seguenti cammini per i diversi agenti:

- Agente *a1*: (1,2), (2,4);
- Agente *a2*: (4,5), (5,2);
- Agente *a3*: (3,4), (4,5), (5,2);

Il valore della funzione obiettivo è *7*, ovvero la somma degli archi percorsi dai vari agenti. Se considerassimo il percorso minimo di ogni agente (come se si muovesse da solo sul grafo) otterremmo i cammini in figura 2:

- (1,2), (2,4) per l'agente *a1*;
- (4,2) per l'agente *a2*;
- (3,4), (4,2) per l'agente *a3*;

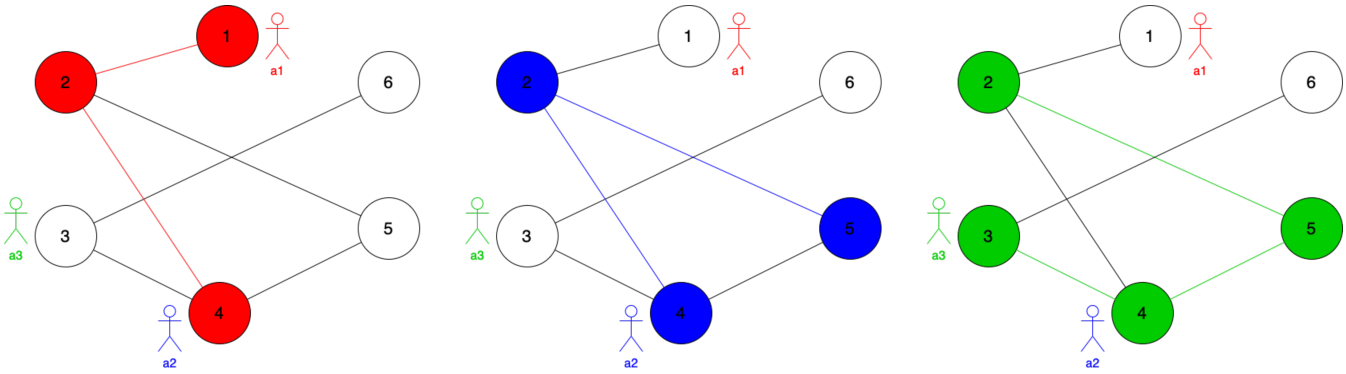

Figure 2: Percorsi svolti dai singoli agenti.

Se gli agenti svolgessero i cammini appena descritti contemporaneamente ci sarebbero due collisioni: gli agenti *a1* e *a2* si troverebbero contemporaneamente sul nodo *2*, mentre gli agenti *a1* e *a3* attraverserebbero (in direzioni opposte) l'arco (2, 4) contemporaneamente. Possiamo quindi osservare come il modello riesca a trovare percorsi diversi (i minimi possibili) per poter evitare tali collisioni.

Modificando il grafo sostituendo l'arco (3,4) con l'arco (2,6), (come di seguito), si ottiene un problema non risolvibile, a meno di collisioni;

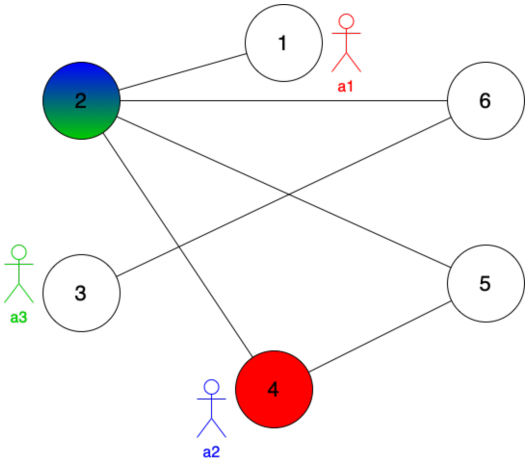

Figure 3: Grafo modificato.

l'agente *a2* ha una solo cammino percorribile che inevitabilmente, avendo supposto che un agente non possa fermarsi ad aspettare su un nodo, porta alla collisione con l'agente *a1* sull'arco (2,6). Esiste anche una collisione tra gli agenti *a2* e *a3* sul nodo *2*: l'agente *a1* all'istante 1 deve necessariamente percorrere l'arco (1,2), questo implica che *a2* debba, per evitarlo, percorrere l'arco (4,5) (unica alternativa possibile); a questo punto *a2* dovrebbe entrare nel vertice *2*, cui però vuole accedere, nello stesso istante, anche l'agente *a3*, che sta percorrendo l'unico cammino possibile per arrivare a destinazione. Il problema risulta quindi irrisolvibile.

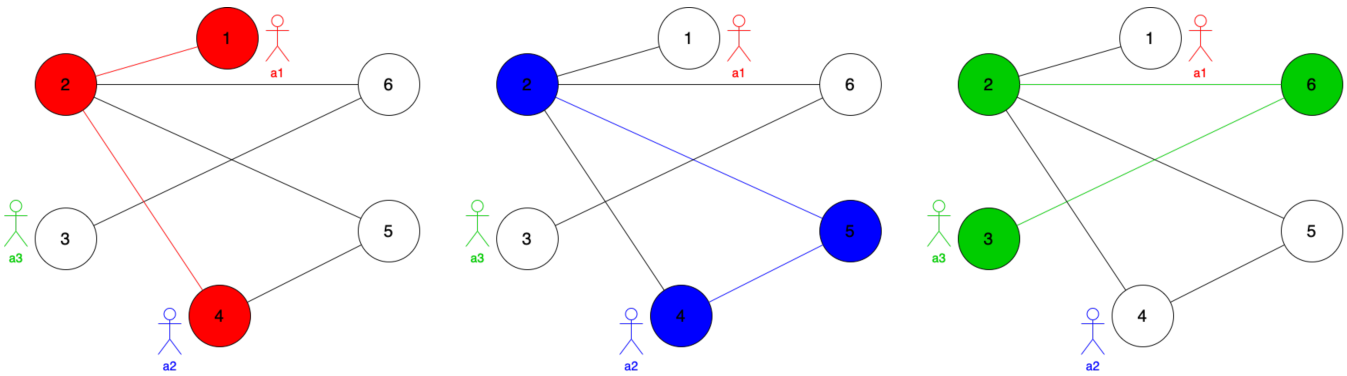

Figure 4: Problema irrisolvibile.

Riferimenti

[1] Roman. Barták, Jirí. Srvancara, and Marek Vlk. Scheduling models for multi-agent path finding. 2017.

[2] `http://yetanothermathprogrammingconsultant.blogspot.com/2016/05/all-different-and-mixed-integer.html`.