Università degli Studi di Cagliari

Network Flows Optimization

Tanker Scheduling Problem

Anno accademico: 2023/24

Docente: Massimo Di Francesco

Studenti:

Lorenzo Boi

Sebastiano Felice Mereu

Indice:

Il problema

Una compagnia di navigazione a vapore ha stipulato un contratto per consegnare merci deperibili tra diverse coppie di origine e destinazione.

Poiché il carico è deperibile, i clienti hanno specificato date di consegna precise entro le quali le spedizioni devono raggiungere le loro destinazioni. I carichi non possono arrivare in anticipo o in ritardo.

La compagnia di navigazione vuole determinare il numero minimo di navi necessarie per rispettare le date di consegna dei carichi.

Per illustrare un approccio di modellazione per questo problema, consideriamo un esempio con quattro spedizioni; ciascuna spedizione è un carico completo con le caratteristiche mostrate nella tabella 1. Ad esempio, come specificato dalla prima riga della tabella, l'azienda deve consegnare un carico disponibile al porto A e destinato al porto C il giorno 3.

Le tabelle 2 e 3, invece, mostrano i tempi di transito per le spedizioni (comprensivi di margini per il carico e lo scarico delle navi) e i tempi di ritorno (senza carico) tra i porti.

Spedizione	Origine	Destinazione	Data di consegna
1	Porto A	Porto C	3
2	Porto A	Porto C	8
3	Porto B	Porto D	3
4	Porto B	Porto C	6

Tabella 1: lista delle spedizioni con caratteristiche

Navi Cariche	C	D
A	3	2
B	2	3

Tabella 2: tempi di trasporto delle navi cariche da un porto all'altro

Navi Scariche	A	B
C	2	1
D	1	2

Tabella 3: tempi di transito delle navi scariche da un porto all'altro

Nella figura 4, invece, viene mostrata una linea temporale rappresentante i tempi di consegna, con indicazione di arrivo e destinazione dei porti per ogni spedizione.

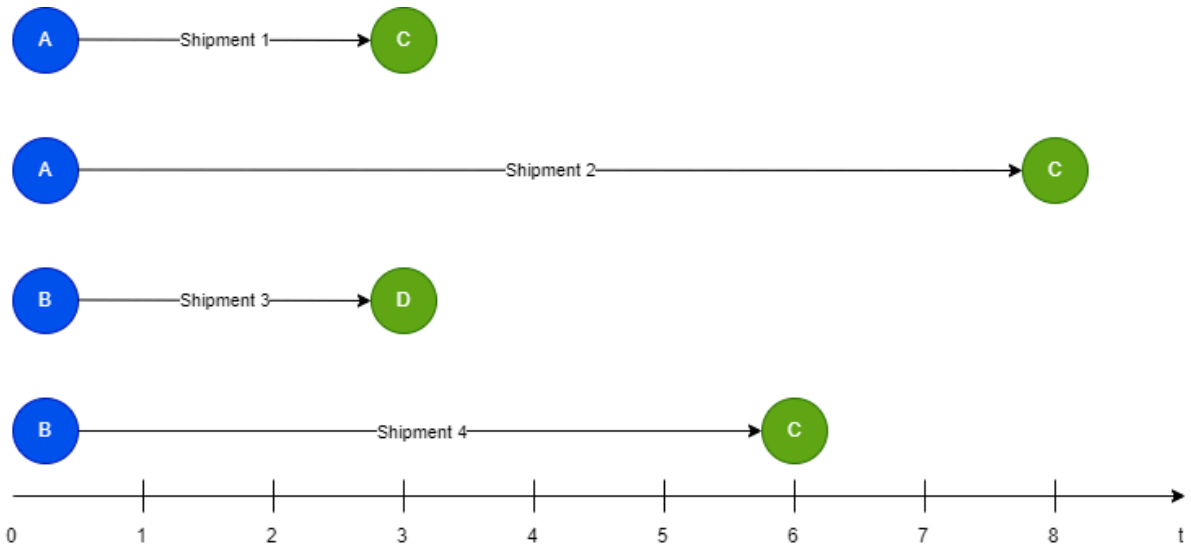

Figura 4: In "blu" i porti di origine, in "verde" i porti di destinazione. I porti di destinazione cadono nel giorno di consegna richiesto.

Costruzione del grafo delle spedizioni

Per risolvere il problema è stato modellato un grafo tenendo conto delle spedizioni, delle date di consegna previste e facendo assunzioni sui tempi di partenza in base ai tempi di carico e scarico delle navi, in particolare, nella costruzione del grafo è stato tenuto conto della compatibilità tra le spedizioni per creare i collegamenti tra i nodi, inoltre, sono stati aggiunti i nodi sorgente e destinazione (s e t) permettendo così di vedere il problema da un punto di vista di minimizzazione del numero dei percorsi da s a t.

Questo tipo di modellazione permette di ricondurre il problema a un tipo specifico, il problema del valore minimo.

Di seguito vengono mostrati nel dettaglio gli step per la costruzione del grafo, a supporto delle considerazioni di compatibilità tra le spedizioni è stata aggiunta una linea temporale.

Step 1: Aggiunta delle spedizioni 1 e 3

Per l'aggiunta di queste è stato tenuto conto del fatto che la spedizione 1 deve essere consegnata al tempo 3, dal porto A al porto C, una nave carica per questo tragitto necessita di 3 slot temporali, perciò dovrà partire al tempo 0.

Lo stesso ragionamento, con gli stessi tempi può essere fatto per la spedizione 3.

Nella figura 5 vengono mostrate le due spedizioni nella linea temporale, i tempi di arrivo e destinazione sono riportati con relativi suffissi "p" e "s"

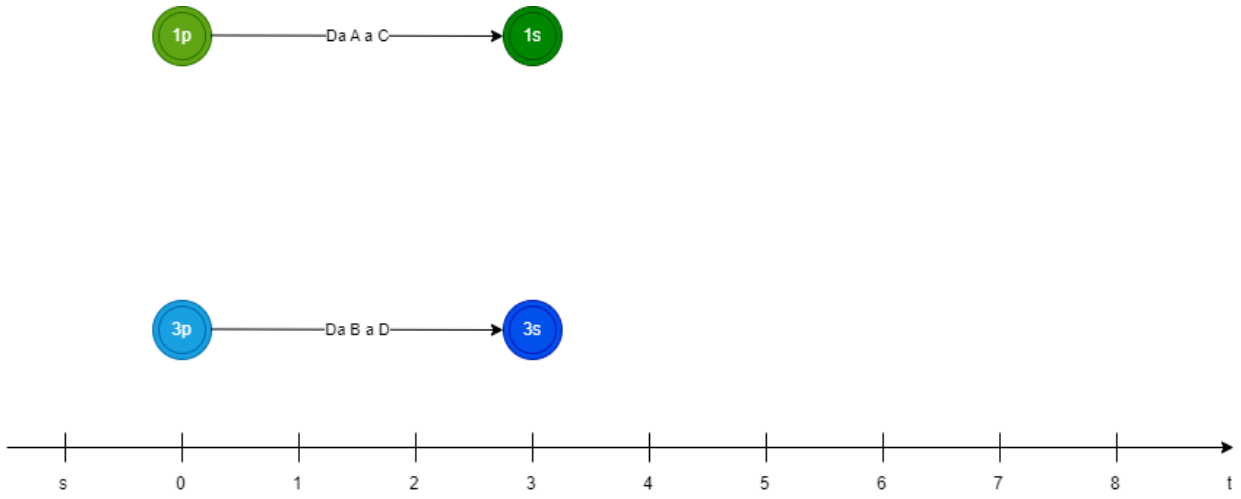

Figura 5: aggiunta Spedizioni 1 e 3

Step 2: Aggiunta delle spedizioni 2 e 4

Per l'aggiunta di queste è stato tenuto conto del fatto che la spedizione 2 deve essere consegnata al tempo 8, dal porto A al porto C, una nave carica per questo tragitto necessita di 3 slot temporali, perciò dovrà partire al tempo 0.

Per quanto riguarda la spedizione 4 invece, questa deve essere consegnata al tempo 6, dal porto B al porto C, la nave carica da B a C necessita di 2 slot temporali, quindi dovrà partire al tempo 4.

Nella figura 6 vengono mostrate le due spedizioni nella linea temporale

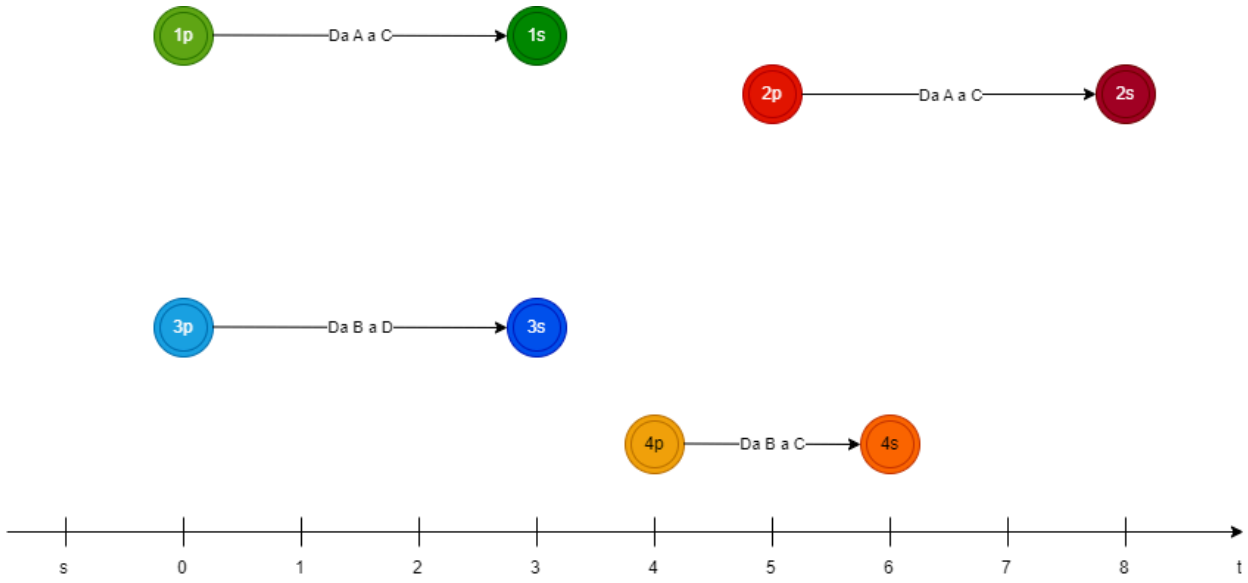

Figura 6: aggiunta Spedizioni 2 e 4

Step 3: Collegamento delle spedizioni in base alla compatibilità

Ora che le spedizioni sono correttamente riportate in base ai tempi di partenza e di arrivo delle navi nei rispettivi porti, è necessario fare delle assunzioni sui possibili collegamenti tra i porti, tenendo conto dei tempi di transito delle navi scariche.

Detto ciò si è tenuto conto del collegamento tra le spedizioni 1 e 3, rispetto alle spedizioni 2 e 4:

- Partenza a fine spedizione 1:
  - Si parte dal Porto C, e si deve arrivare alla spedizione 2 che parte dal Porto A, la nave scarica da C ad A necessita di *2 slot temporali*, quindi il collegamento è *possibile*.
  - Si parte dal Porto C, e si deve arrivare alla spedizione 4 che parte dal Porto B, la nave scarica da C a B necessita di *1 slot temporale*, quindi il collegamento è *possibile*.
- Partenza a fine spedizione 3:
  - Si parte dal Porto D, e si deve arrivare alla spedizione 2, che parte dal Porto A, la nave scarica da D ad A necessita di *1 slot temporale*, quindi il collegamento è *possibile*.
  - Si parte dal Porto D, e si deve arrivare alla spedizione 4, che parte dal Porto B, la nave scarica da D ad B necessita di *2 slot temporali*, quindi il collegamento *non è possibile*.

Nella figura 7 vengono mostrate le spedizioni con i rispettivi collegamenti.

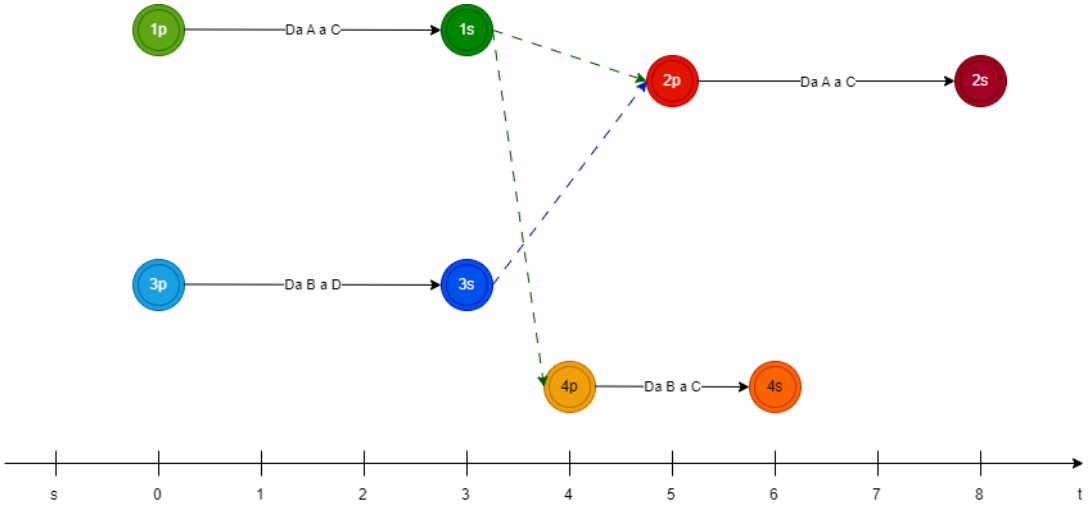

Figura 7: collegamenti spedizioni

Step 4: Aggiunta nodi s e t

L'ultimo step della costruzione del grafo è quello di aggiunta dei nodi s e t con i rispettivi collegamenti, a s sono stati collegati i nodi di "partenza" delle spedizioni (p) e a t sono stati collegati i nodi di arrivo delle spedizioni (s). Il grafo finale, con i collegamenti s e t è mostrato nella figura 8.

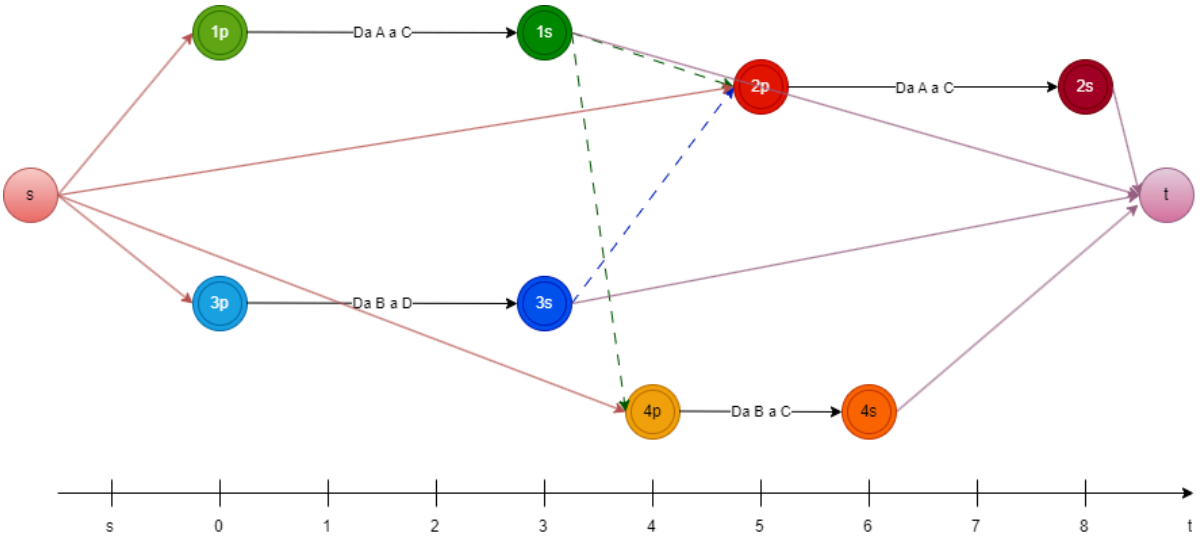

Figura 8: nodi s e t

Formulazione matematica

Siano:

- N l'insieme dei nodi, compresi i nodi speciali s e t
- A l'insieme degli archi, che rappresentano i possibili percorsi tra i nodi
- C l'insieme dei nodi di carico, i nodi dove la nave parte verso una destinazione
- S l'insieme dei nodi di scarico, i nodi dove la nave scarica la merce, ovvero le destinazioni

Lista variabili decisionali:

- f_{ij} è una variabile binaria che vale 1 se la è utilizzata per fare il viaggio dall'arco i a j

Funzione obiettivo:
minimizzare il numero totale di navi utilizzate, quindi minimizzare il numero di archi che partono dal nodo sorgente *s* che può essere espresso come:

$$\min \sum_{j \in C} f_{sj}$$

Vincoli:

1) Almeno un percorso deve partire dal nodo sorgente *s* verso un nodo di carico

$$\sum_{j \in C \setminus \{s\}} f_{sj} \geq 1$$

2) Vincolo di conservazione del flusso, quello che entra in un nodo deve anche uscire, per tutti i nodi tranne s (e anche t):

$$\forall (i,j) \in N/\{s,t\} : \sum_{j \in \delta(i)+} f_{ij} - \sum_{j \in \delta(i)-} f_{ji} = 0$$

Implementazione

Dopo aver analizzato il problema, ricavato la formulazione matematica della funzione obiettivo e dei vincoli, il modello è stato effettivamente implementato.

Si sono usate due librerie python: **docplex e cplex**.
Il grafo è stato rappresentato da un dizionario, ogni chiave rappresenta un nodo e ogni valore rappresenta una lista di possibili destinazioni a partire da quel nodo.

```
grafo = {
    's':['1p','2p','3p','4p'],
    '1p':['1s'],
    '2p':['2s'],
    '3p':['3s'],
    '4p':['4s'],
    '2s':['t'],
    '4s':['t'],
    '3s':['2p','4p','t'],
    '1s':['4p','2p','t'],
}
```

Figura 9: rappresentazione grafo nel dizionario

Successivamente, dopo aver inizializzato il modello, abbiamo creato le variabili decisionali (binarie) che assumono valore 1 se quell'arco è stato scelto per il percorso della nave, altrimenti zero.

```
# Variabili di decisione: indicano se un arco è selezionato
percorsi = {(u, v): modello.binary_var(name=f'path_{u}_{v}') for u in grafo.keys() for v in grafo[u]}
```

Figura 10: variabili decisionali

La funzione obiettivo è definita come:

```
modello.minimize(modello.sum(percorsi['s', v] for v in grafo['s']))
```

Figura 11: implementazione funzione obiettivo

La funzione obiettivo andrà a minimizzare gli archi che partono dal nodo sorgente (s) , che sono stati aggiunti al grafo, il numero degli archi che partono da questo particolare nodo saranno uguali al numero di navi utilizzate per i nostri trasporti.

Alla funzione obiettivo seguono i 3 vincoli:

```
# Vincolo 1: almeno un percorso deve partire da 's'
vincolo_s = modello.sum(percorsi['s', v] for v in grafo['s']) >= 1
modello.add_constraint(vincolo_s, ctname='at_least_one_path_from_s')
```

```python
# Vincolo 2: ogni arco deve essere utilizzato (tranne quelli che partono da s e arrivano in t)
for u in grafo.keys():
    for v in grafo[u]:
        if v!='t':
          vincolo = modello.sum(percorsi[u, v] for u in grafo.keys() if v in grafo[u]) == 1
          modello.add_constraint(vincolo, ctname=f'edge_usage_{u}_{v}')
```

```python
# Vincolo 3: solo un arco esce da ogni nodo diverso da 's'
for nodo in grafo.keys():
    if nodo != 's':
        vincolo_uscita = modello.sum(percorsi[nodo, v] for v in grafo[nodo]) == 1
        modello.add_constraint(vincolo_uscita, ctname=f'only_one_edge_out_{nodo}')
```

Figure 12, 13, 14: vincoli

All'implementazione vera e propria del modello sono state aggiunte delle funzioni di supporto:

- *stampa_risultati(modello,percorsi):* Utile allo scopo di ricavare dalla soluzione il numero di path e gli archi coinvolti.
- *trova_percorsi(modello,percorsi,grafo):* Utile per poter ricostruire i path a partire dagli archi selezionati.

Codice Implementazione: https://github.com/loriboi/nfo_tanker_scheduling_problem

Risultati ottenuti

Una volta terminata l'esecuzione dell'algoritmo per la risoluzione del modello, è stato trovato che nel caso analizzato nella tesina sono necessarie 2 navi, quindi 2 path da s a t:

- Primo path: si parte dalla spedizione 3 per arrivare alla spedizione 2, quindi si parte dal porto B, e la nave consegnerà al porto D, poi si va al porto A con consegna al porto C
- Secondo path: si parte dalla spedizione 1, per arrivare alla spedizione 4, quindi si parte dal porto A, e la nave consegnerà al porto C, successivamente si va al porto B con consegna al porto C.

I risultati ottenuti utilizzando l'implementazione via codice vista in precedenza sono mostrati nelle figure 15 e 16:

```
path: 2.0
Arco s a 1p è selezionato
Arco s a 3p è selezionato
Arco 1p a 1s è selezionato
Arco 2p a 2s è selezionato
Arco 3p a 3s è selezionato
Arco 4p a 4s è selezionato
Arco 2s a t è selezionato
Arco 4s a t è selezionato
Arco 3s a 2p è selezionato
Arco 1s a 4p è selezionato
```

Figura 15: numero di path scelti e archi selezionati

```
Percorsi trovati: [['s', '1p', '1s', '4p', '4s', 't'], ['s', '3p', '3s', '2p', '2s', 't']]
```

Figura 16: ricostruzione dei path

Nelle figure seguenti la soluzione nel dettaglio con i path trovati.

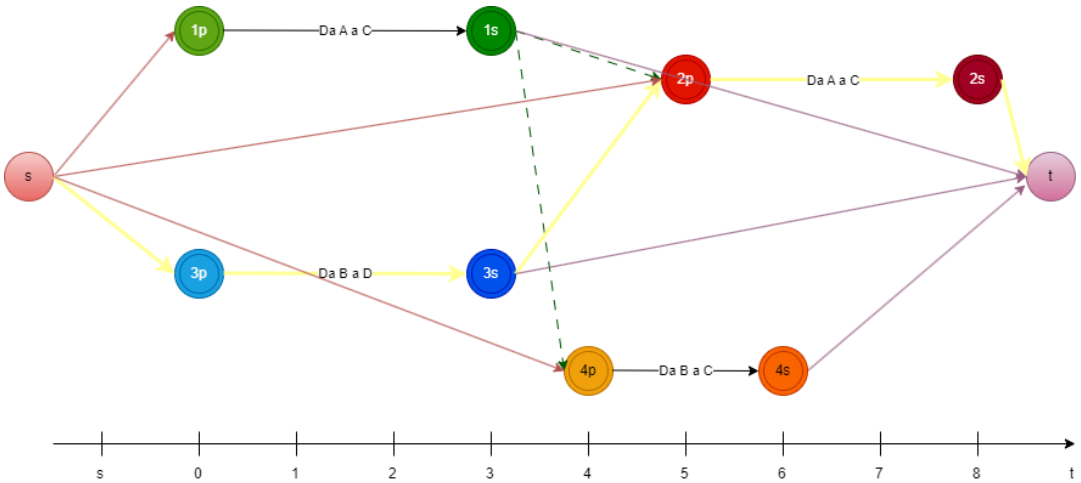

Figura 17: tragitto prima nave

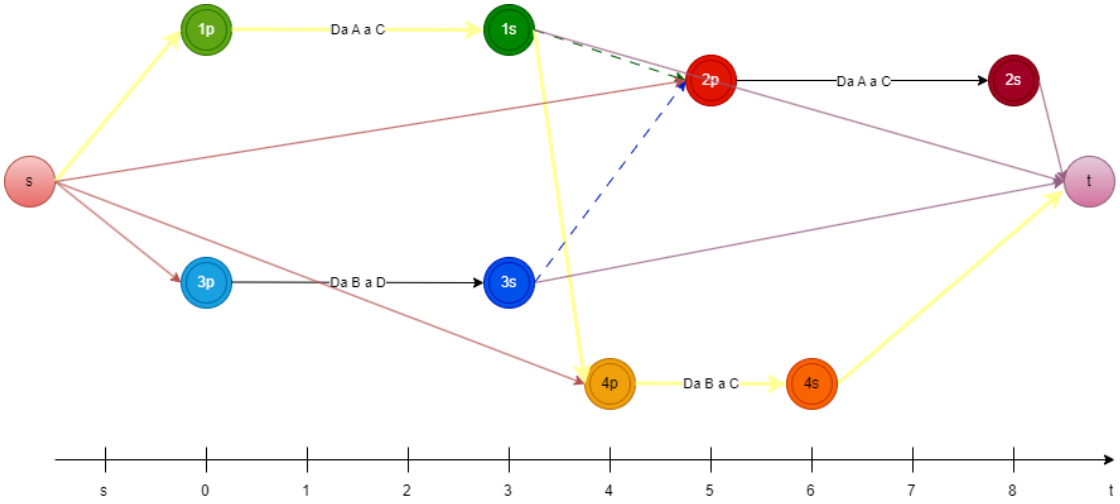

Figura 18: tragitto seconda nave

Conclusioni e prossime implementazioni

Il modello testato su poche spedizioni si comporta bene, infatti fornisce i risultati attesi facilmente verificabili.

Non è stato testato su dataset di grande dimensione, ma ci si attende che continui a funzionare in tempi ragionevoli.

Una possibile integrazione a livello di implementazione potrebbe essere quella di creare un sistema per poter generare il grafo delle spedizioni a partire da un dataset di queste con rispettivi tempi di trasporto e transito delle navi.