Tor Vergata

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica

Tesi di laurea

Consegna efficiente di messaggi tramite l'utilizzo di agenti mobili: il caso singola sorgente

Candidato:
Leonardo Tamiano
Matricola 0227580

Relatore:
Luciano Gualà

Anno Accademico 2017–2018

Indice

Capitolo 1

Introduzione

I recenti progressi tecnologici nel campo della robotica stanno facilitando la produzione di massa e l'utilizzo di robot mobili, che noi chiameremo *agenti mobili*, o semplicemente *agenti*. Questi agenti possono essere utilizzati in modo automatico senza intervento umano in svariati contesti. Al fine di far funzionare correttamente questi team di agenti, risulta necessario risolvere una serie di problemi algoritmici, che dipendono dal particolare contesto in cui gli agenti vengono utilizzati.

In questa tesi consideriamo il problema di spostare, o consegnare, un certo insieme di oggetti, o messaggi, da dei punti di partenza a dei punti di destinazione. In questo contesto dunque gli agenti mobili sono in grado di raccogliere i messaggi nei punti di partenza (sorgenti), e muoversi all'interno della rete per andarli a consegnare nei rispettivi punti di destinazione. Durante la consegna, gli agenti consumano una quantità di energia proporzionale alla distanza percorsa. Siamo quindi interessati a capire quali sono le scelte più efficienti, in termine di energia totale consumata, che i vari agenti devono eseguire al fine di consegnare tutti i messaggi in modo corretto.

In generale non tutti gli agenti sono identici; alcuni potrebbero essere più efficienti rispetto ad altri, a seconda della particolare tecnologia utilizzata. Assumiamo dunque che ad ogni agente è associato un *peso*, che indica la quantità di energia consumata dall'agente per ogni unità di distanza percorsa. Inoltre, gli agenti potrebbero partire da posizioni diverse. Dunque, potrebbe risultare che la scelta più efficiente per un agente sia quella di raccogliere un messaggio, per poi rilasciarlo ad un altro agente in una posizione intermedia.

Trovare una soluzione ottimale che minimizza l'energia totale consumata consiste nello stabilire quali sono le mosse compiute dagli agenti, ovvero capire quali messaggi, e in che ordine, ciascun agente deve raccogliere, e dove questi messaggi devono essere rilasciati. Noi studieremo questa problematica basandoci sulla formalizzazione presente in [1], dove è definito un problema, chiamato **WeightedDelivery**, che modella l'ambiente in cui si muovono gli agenti come un grafo pesato, in cui i pesi sugli archi rappresentano distanze e le posizioni dei messaggi e degli agenti sono nodi.

In [1] il problema del WeightedDelivery è analizzato da diversi aspetti. Fra i tanti contributi è importante citare che gli autori hanno trovato un algoritmo polinomiale per il caso in cui c'è un solo messaggio da consegnare, ma dimostrano che il problema di capire

come consegnare tutti i messaggi minimizzando il consumo energetico totale, in generale, è un problema NP-hard. Viene inoltre studiata come cambia la qualità di una soluzione se si impone che gli agenti non si possono scambiare i messaggi tra loro. Infine, è descritto un algoritmo di approssimazione per il problema generale.

In questa tesi, noi ci concentriamo su un caso speciale, ma comunque interessante, in cui tutti i messaggi da consegnare si trovano inizialmente nello stesso nodo di partenza, da noi chiamato *sorgente*. Tale caso è stato studiato in due contesti diversi, a seconda che i pesi degli agenti siano uniformi o meno. Per il caso in cui i pesi degli agenti sono uniformi, ovvero quando tutti gli agenti hanno lo stesso peso, abbiamo trovato un algoritmo greedy che risolve in modo esatto il problema in tempo polinomiale. Se invece i pesi non sono uniformi, abbiamo trovato un algoritmo che ottiene una 2-approssimazione del problema generale.

La tesi è strutturata nel seguente modo: nel Capitolo 2 è presente la formalizzazione matematica per il problema del WeightedDelivery. Il Capitolo 3 descrive lo stato dell'arte sul problema, mentre i risultati originali ottenuti per il caso a singola sorgente sono trattati nel Capitolo 4. Infine, il Capitolo 5 fornisce alcune direzioni di ricerca rimaste ancora inesplorate.

Capitolo 2

Descrizione del modello

Una generica istanza del problema è composta dai seguenti componenti: un grafo non diretto e pesato $G = (V, E)$; k agenti, denotati $a_1, ..., a_k$; ed m messaggi, denotati $m_1, ...m_m$.

Ad ogni arco del grafo $e \in E$ è associato un valore l_e, che rappresenta la lungehzza dell'arco.

Ad ogni agente a_i, sono associate due cose: un nodo $p_i \in V$ del grafo G, che rappresenta il nodo di partenza dell'agente, e un peso w_i, che rappresenta l'efficienza che l'agente ha nel muoversi all'interno del grafo. Quando l'agente a_i percorre l'arco e, il costo totale per effettuare tale movimento è $l_e \cdot w_i$.

Un messaggio m_i è definito da una coppia di nodi (s_i, t_i), dove s_i rappresenta la sorgente del messaggio, ovvero il nodo in cui il messaggio si trova inizialmente, e t_i rappresenta il nodo di destinazione in cui il messaggio deve essere consegnato.

Infine, definiamo la lunghezza di un cammino semplice π come la somma delle lunghezze degli archi che compongo il cammino, e denotiamo con $d_G(u, v)$ la distanza tra u e v nel grafo G, definita come la lunghezza di un qualsiasi cammino minimo tra u e v nel grafo G.

Gli agenti si possono muovere nel grafo, e possono raccogliere e depositare i messaggi nei nodi del grafo al fine di consegnare tutti i messaggi nelle rispettive destinazioni. Nel modello generale presente in [1] è presente anche il parametro κ, che specifica la *capacità* degli agenti, intesa come il numero di messaggi che ciascun agente può trasportare in un dato istante. In ogni caso, dato che molto spesso si studiano istanze in cui $\kappa \in \{1, \infty\}$, e dato che molti risultati utili sono ottenuti nel caso $\kappa = 1$, noi lavoreremo sempre con istanze in cui $\kappa = 1$. Dunque, nel nostro modello ogni agente può trasportare al più un messaggio in un dato istante.

Denotiamo con d_i la distanza totale percorsa dall'agente a_i per la consegna dei messaggi. Il problema **WeightedDelivery** è dunque il problema di ottimizzazione del costo totale $\sum_{i=1}^{k} w_i \cdot d_i$ necessario per la consegna di tutti i messaggi nelle rispettive destinazioni.

Al fine di descrivere una possibile soluzione del problema si utilizza il concetto di *schedule*. Uno schedule S descrive le azioni dei vari agenti tramite una sequenza ordinata di quadruple. Ogni quadrupla rappresenta una particolare azione che un agente può compiere. Le azioni sono di due tipologie: azioni di *pick-up* (raccolta), e azioni di *drop-off* (rilascio). Una azione di pick-up è della forma $(a_i, p, m_i, +)$ e deve essere interpretata nel seguente modo: "l'agente a_i si sposta dal nodo in cui si trova al nodo p e *raccoglie* il messaggio m_i"; una azione di drop-off invece è della forma $(a_i, q, m_i, -)$ e deve essere interpretata nel seguente modo: "l'agente a_i si sposta dal nodo in cui si trova al nodo q e *rilascia* il messaggio m_i nel nodo q". Notiamo che uno schedule S codifica implicitamente tutti gli spostamenti degli agenti, ed è dunque possibile calcolare facilmente il costo totale per la consegna $COST(S) := \sum_{i=1}^{k} w_i \cdot d_i$.

Dato uno schedule S, denotiamo con $S|_{a_j}$ tutte le azioni eseguite dall'agente a_j nello schedule S, mentre con $S|_{m_i}$ denotiamo tutte le azioni riguardanti la raccolta (pick-up) o il rilascio (drop-off) del messaggio m_i all'interno dello schedule S. Uno schedule S è detto *feasible* (ammissibile) se ogni azione di pick-up della forma $(\_, p, m_i, +)$, con $p \neq s_i$, è preceduta da una azione di drop-off della forma $(\_, p, m_i, -)$, e se tutti i messaggi sono consegnati.

Capitolo 3

Stato dell'arte

In questo capitolo è presente una breve descrizione dei risultati ottenuti in [1], e qualche nota riguardante il concetto di *collaborazione* in uno schedule.

3.1 Risultati noti

Il primo risultato ottenuto in [1] è positivo, e fa vedere che il problema del WeightedDelivery con un solo messaggio, $m = 1$, può essere risolto polinomialmente riducendolo al problema del calcolo di uno cammino minimo su un grafo ausiliario costruito utilizzando l'algoritmo "all pair shortest paths" di *Floyd Warshall* [2, 4].

Successivamente i ricercatori utilizzano il problema generale del WeightedDelivery per definire tre problemi correlati ad esso, che sono il problema della *collaborazione* (collaboration), che vuole trovare il modo migliore per consegnare ciascun messaggio utilizzando più agenti; il problema della *pianificazione* (planning), che vuole trovare l'ordine migliore in cui ciascun agente consegna i propri messaggi; e per finire il problema della *coordinazione* (coordination), che vuole trovare il modo migliore per assegnare i messaggi agli agenti.

Per ogni problema menzionato l'articolo dimostra vari risultati, alcuni positivi e altri negativi. Infine, utilizzando principalmente i risultati ottenuti dallo studio del problema della collaborazione, insieme all'idea su cui si basa l'algoritmo di Kruskal [3], l'articolo descrive un algoritmo che ottiene una $4 \cdot \max \frac{w_i}{w_j}$ approssimazione del problema WeightedDelivery.

3.2 Note sulla collaborazione

Notiamo che, tra i tre problemi correlati al problema WeightedDelivery (collaborazione, pianificazione, coordinazione), il più importante e interessante è il problema della collaborazione. Dato uno schedule S, diciamo che in S c'è collaborazione se esiste almeno un messaggio m_i che, nel corso del suo tragitto dalla sorgente s_i alla destinazione t_i, viene trasportato da almeno due agenti diversi. Viceversa, diciamo che in S non c'è collabora-

zione se, per ogni messaggio m_i, esiste un solo agente che trasporta m_i dalla sorgente alla destinazione.

Nel contesto della collaborazione, in [1] viene dimostrato (Teorema 4, Corollario 5) che non è possibile definire un algoritmo che è in grado di risolvere il problema del Weighted-Delivery senza collaborare ottenendo un qualcosa di migliore di una 2-approssimazione. Questo vuol dire che, qualunque algoritmo definiamo, se tale algoritmo non fa collaborare gli agenti tra loro, allora esisterà sempre almeno una istanza in cui la soluzione trovata dall'algoritmo (soluzione che non collabora) si discosta di un fattore 2 dalla soluzione ottima (soluzione che collabora). Sempre in [1] viene poi dimostrato che, non collaborando, il costo della soluzione trovata è al più il doppio del costo della soluzione ottima. Unendo questi due risultati dunque l'articolo dimostra che, generalmente, non collaborando si paga un fattore di approsimazione pari a 2.

Un'ulteriore restrizione che possiamo fare sul modo in cui gli agenti trasportano i messaggi è la seguente: possiamo imporre che, dopo che un agente a_j raccoglie un messaggio m_i, tale agente non potrà più rilasciare il messaggio in un nodo v diverso dal nodo di destinazione t_i. In questo caso diciamo che non ci sono *drop-offs intermerdi*, e gli schedule che rispettano questa restrizione sono detti *schedule ristretti*. Dato che per collaborare necessariamente bisogna effettuare un drop-off intermedio, abbiamo che quest'ultima restrizione implica la non collaborazione, ma non vale il viceversa. Formalmente S è uno schedule ristretto se vale il seguente predicato

$$\forall i \exists j : S|_{m_i} = (a_j, s_i, m_i+), (a_j, t_i, m_i, -)$$

In [1] viene dimostrato che, per le istanze in cui κ [1] $\in \{1, \infty\}$, esiste sempre uno schedule ristretto, ovvero uno schedule in cui non ci sono drop-offs intermerdi, il cui costo non supera mai di un fattore 2 il costo della soluzione ottima ottenuta collaborando.

[1] Ricordiamo che κ è la capacità di capienza degli agenti, intesa come il numero massimo di messaggi che un agente può trasportare in un dato istante. Nel nostro modello abbiamo imposto che $\kappa = 1$.

Capitolo 4

Il caso a singola sorgente: due nuovi algoritmi

Nel nostro lavoro ci siamo concentrati sulle istanze del problema WeightedDelivery che rispettano la restrizione aggiuntiva da noi chiamata *same message source* (**SMS**). Tale restrizione impone che la posizione di partenza di ogni messaggio è la stessa, ovvero che $s_i = s$ per $i = 1, ..., m$. Oltre a questa restrizione abbiamo lavorato con un'altra restrizione, da noi chiamata *uniform weight* (**UW**), che impone che i pesi degli agenti sono uniformi, ovvero che $w_i = w$ per $i = 1, ..., k$.

4.1 Un algoritmo esatto per il caso uniforme

Sotto le restrizioni **SMS+UW** possiamo definire il seguente algoritmo greedy:

Finché ci sono messaggi da consegnare, consegna il messaggio la cui destinazione è la più vicina alla sorgente s, utilizzando l'agente più vicino alla sorgente s.

Iniziamo l'analisi notando che tale algoritmo è polinomiale e produce chiaramente uno schedule ammissibile. Per quanto riguarda l'ottimalità, procederemo in due passi: prima dimostriamo che sotto le restrizioni **SMS+UW** è sempre possibile trovare uno schedule ristretto che è anche uno schedule ottimo; poi dimostriamo che l'algoritmo descritto calcola sempre uno schedule ristretto ottimo per l'istanza data. Unendo questi due risultati segue che l'algoritmo greedy è un algoritmo polinomiale esatto per il problema.

4.1.1 Schedule ristretti e schedule ottimi sotto SMS+UW

Andiamo adesso a far vedere che, per le istanze che soddisfano le restrizioni **SMS+UW**, è sempre possibile trovare uno schedule ristretto S^R che è anche uno schedule ottimo.

Proprietà 4.1.1. Sia I una istanza del problema WeightedDelviery che rispetta la restrizione **UW** e tale che la capacità degli agenti è unitaria, e sia S_{OPT} uno schedule ottimo per tale istanza. Allora esiste sempre uno schedule S tale che 1) S è ottimo, e 2) S non fa collaborare gli agenti tra loro.

Dimostrazione. Sia m un messaggio che in S_{OPT} viene consegnato facendo collaborare più agenti tra loro, e siano u_1, u_2, ..., u_h i nodi intermedi in cui il messaggio m viene scambiato fra gli agenti. Siano a_i e a_j gli agenti che si scambiano il messaggio m nel nodo intermedio u_1, con a_i che lascia il messaggio a a_j. Andiamo adesso a definire un nuovo schedule, lo schedule S', nel seguente modo: S' è uguale ad S fino all'azione in cui l'agente a_i rilascia il messaggio m nel nodo intermedio u_1, ovvero gli schedule S' e S sono uguali fino alla quadrupla (compresa quest'ultima) $(a_i, u_1, m, -)$. A partire dalla quadrupla successiva, in S' tutte le azioni che in S venivano eseguite dall'agente a_i vengono eseguite dall'agente a_j, e viceversa, tutte le azioni che in S venivano eseguite dall'agente a_j, in S' vengono eseguite dall'agente a_i. Tutte le azioni eseguite in S da agenti diversi da a_i e a_j rimangono invariate nello schedule S'.

Notiamo che, effettuando questo scambio di traiettoria, non stiamo cambiando la distanza totale percorsa dagli agenti. Dato che i pesi sono uniformi, questo implica poi che anche il costo tra gli schedule non cambia. Formalmente, vale il seguente

$$COST(S) = \sum_{i=1}^{k} w_i \cdot d_i = \sum_{i=1}^{k} w \cdot d_i = w \cdot D(S)$$

dove $D(S)$ rappresenta la distanza totale percosa dagli agenti nello schedule S. Per come abbiamo costruito lo schedule S' abbiamo che $D(S) = D(S')$, e dunque $COST(S) = COST(S')$. Inoltre, dato che la capacità degli agenti è unitaria, il nuovo schedule S' rimane ancora feasible in quanto le capacità degli agenti rispetto alle traiettoria eseguite sono le stesse dello schedule S.

Osserviamo adesso che S', per consegnare il messaggio m, effettua una collaborazione in meno rispetto allo schedule S. Ripetendo questo ragionamento per ogni nodo intermedio rimanente $u_2, ..., u_h$, riusciamo ad ottenere uno schedule S'' che consegna m senza collaborare e senza peggiorare il costo. Infine, ripetendo nuovamente questo ragionamento per ogni messaggio, siamo in grado di costruire uno schedule S che consegna ogni messaggio senza collaborare e senza peggiorare il costo dello schedule iniziale. □

Proprietà 4.1.2. Sia I una istanza del problema WeightedDelivery che rispetta le restrizioni **UW+SMS**, e sia S_{OPT} uno schedule ottimo per tale istanza. Allora esiste sempre uno schedule S tale che 1) S è ottimo, e 2) S non esegue drop-offs intermerdi.

Dimostrazione. Dal risultato precedente possiamo assumere senza perdità di generalità che nello schedule S_{OPT} gli agenti non collaborano tra loro. Supponiamo poi che nello schedule S_{OPT} esiste un agente a_j che esegue almeno un drop-off intermedio. Consideriamo il primo drop-off eseguito dall'agente a_j, e supponiamo che viene effettuato nel nodo u durante la consegna del messaggio m_i. Andiamo adesso a far vedere che è sempre possibile rimuovere questo drop-off intermedio senza peggiorare il costo dello schedule.

Ricordiamo che lavoriamo sotto la restrizione **SMS**, e che gli agenti hanno una capacità unitaria. Queste restrizioni, combinate con il fatto che il drop-off considerato è il primo eseguito dall'agente a_j, implicano che, dopo aver rilasciato il messaggio m_i nel nodo u, l'agente a_j, se vuole consegnare altri messaggi, deve necessariamente ritornare nel nodo

s. Inoltre, dato che non c'è collaborazione, il messaggio m_i deve necessariamente essere consegnato dall'agente a_j, e quindi, ad un certo punto, a_j deve per forza ritornare nel nodo u per raccogliere il messaggio m_i.

Definiamo ora lo schedule S. Lo schedule S è uguale allo schedule S_{OPT} tranne per le azioni compiute da a_j. Per quanto riguarda come cambiamo le azioni di a_j in S, consideriamo prima le azioni che a_j esegue nello schedule S_{OPT}:

$$S_{OPT}|_{a_j} = S_A, (a_j, s, m_i, +), (a_j, u, m_i, -), S_B, (a_j, u, m_i, +), S_C$$

dove con S_A rappresentiamo le azioni compiute dall'agente a_j prima di raccogliere il messaggio m_i dalla sorgente, con S_B rappresentiamo le azioni compiute dall'agente a_j tra il rilascio e la raccolta del messaggio m_i nel nodo intermedio u, e con S_C rappresentiamo le azioni dell'agente a_j dopo aver raccolto il messaggio m_i. Notiamo che, da quanto detto prima, $S_B = [(a_j, s, m_l, +), S_D]$, ovvero a_j, dopo aver rilasciato il messaggio m_i, va a raccogliere un altro messaggio dalla sorgente s e continua a fare delle altre azioni S_D. In modo analogo abbiamo che $S_C = [(a_j, q, m_i, -), S_E]$, in cui però il nodo q è un qualsiasi nodo del grafo. La traiettoria di a_j nello schedule S è quindi modificata nel seguente modo

$$S|_{a_j} = S_A, S_B, (a_j, s, m_i, +), S_C$$

Notiamo che il nuovo schedule resta ammissibile, in quanto nessuna azione in S_B va a raccogliere il messaggio m_i dal nodo u. Per quanto riguarda il costo, indicando con w_a l'ultimo nodo visitato da a_j alla fine delle azioni in S_A e con w_b l'ultimo nodo visitato da a_j alla fine delle azioni in S_B, abbiamo che l'agente a_j nello schedule S_{OPT} ha il seguente costo:

$$w_j \cdot [COST(S_A) + d_G(w_a, s) + d_G(s, u) + d_G(u, s) + COST(S_D) + d_G(w_b, u) + d_G(u, q) + COST(S_E)]$$

mentre nello schedule S lo stesso agente ha il seguente costo:

$$w_j \cdot [COST(S_A) + d_G(w_a, s) + COST(S_D) + d_G(w_b, s) + d_G(s, q) + COST(S_E)]$$

Osservando che $d_G(w_b, s) \leq d_G(w_b, u) + d_G(u, s)$, e che $d_G(s, q) \leq d_G(s, u) + d_G(u, q)$, abbiamo che $COST(S_{OPT}|_{a_j}) \geq COST(S|_{a_j})$. Dunque, rimuovendo il primo drop-off effettuato dall'agente a_j, non abbiamo peggiorato il costo.

Ripetendo questo ragionamento su ogni drop-offs successivo eseguito da a_j, e su ogni agente a_j, siamo in grado di ottenere uno schedule S^* che è ottimo e che non esegue drop-off intermedi. □

4.1.2 L'algoritmo greedy calcola uno schedule ristretto ottimo

Sia S_A^R lo schedule ristretto ritornato applicando la regola greedy all'istanza I, e sia S_{OPT}^R un qualsiasi schedule ristretto ottimo per l'istanza I. Per brevità scriveremo sempre S_A e S_{OPT}, omettendo il subscript R.

Vogliamo dimostrare che applicando la regola greedy otteniamo sempre uno schedule ristretto ottimo, ovvero che $COST(S_A) = COST(S_{OPT})$. Per ottenere questo risultato utilizzeremo una argomentazione di tipo *exchange argument*.

Dato che trattiamo solamente di schedule ristretti, nella dimostrazione usiamo il fatto che, a partire da un qualsiasi schedule ristretto S_{OPT}, è sempre possibile riordinarlo per ottenere uno schedule S'_{OPT}, non peggiorando il costo e tale che il nuovo schedule ha la seguente forma: ad ogni azione di pick $(a_j, s, m_i, +)$ segue direttamente la relativa azione di drop-off $(a_j, t_i, m_i, -)$. Notiamo che questo riordinamento delle azioni non cambia il costo della soluzione in quanto 1) la capacità degli agenti è unitaria, e dunque dopo aver preso un messaggio un agente non ne può prendere altri e 2) gli agenti non fanno dei drop-off durante la consegna del messaggio. Lo schedule S'_{OPT} sarà chiamato *schedule ristretto in forma canonica.*

Al fine di dimostrare l'ottimalità dell'algoritmo greedy utilizzeremo la seguente proprietà, che ci permette di ordinare in un modo particolare l'ordine in cui un agente a_j consegna i suoi messaggi.

Proprietà 4.1.3. Sia I una istanza del problema WeightedDelivery che rispetta la restrizioni **SMS**, e sia S un qualsiasi schedule ristretto in forma canonica per tale istanza. Allora esiste sempre uno schedule S' tale che $COST(S') \leq COST(S)$, e tale che per ogni agente a_j in S', i messaggi in $S'|_{a_j}$ sono consegnati rispetto all'ordine delle distanze tra la destinazione del messaggio e la sorgente.

Dimostrazione. Supponiamo che nello schedule S esiste un agente a_j per cui i messaggi in $S|_{a_j}$ non sono consegnati rispetto all'ordine delle distanze tra la destinazione del messaggio e la sorgente. Ovvero supponiamo che l'agente a_j nello schedule S consegna prima il messaggio m_i e subito dopo il messaggio m_j, con $d_G(s, t_i) \geq d_G(s, t_j)$. Notiamo che nello schedule S l'agente a_j incorre nel seguente costo

$$COST(S|_{a_j}) = C_B + d_G(a_j, s) + d_G(s, t_i) + d_G(t_i, s) + d_G(s, t_j) + C_A$$

dove con C_B indichiamo il costo generato da a_j in $S|_{a_j}$ per la consegna di tutti i messaggi prima del messaggio m_i; con $d_G(a_j, s)$ indichiamo la distanza tra la posizione dell'agente a_j in quel particolare momento della consegna–che dipende dalle azioni precedenti presenti in $S|_{a_j}$–e la sorgente s; e con C_A indichiamo il costo generato da a_j in $S|_{a_j}$ per la consegna di tutti i messaggi dopo il messaggio m_j.

Andiamo adesso a definire lo schedule S', uguale allo schedule S tranne per il fatto che in S' l'agente a_j consegna prima il messaggio m_j e poi il messaggio m_i. Il costo di a_j nello schedule S' è pari a

$$COST(S'|_{a_j}) = C_{B'} + d_G(a_j, s) + d_G(s, t_j) + d_G(t_j, s) + d_G(s, t_i) + C_{A'}$$

dove $C_{B'}$ e $C_{A'}$ sono i costi analoghi, rispetto allo schedule S', ai costi C_B e C_A rispetto allo schedule S. Notiamo che $C_{B'} = C_B$, in quanto in entrambi gli schedule l'agente a_j non cambia le sue azioni prima della consegna del messaggio m_i.

Ora, se in S l'agente a_j consegna altri messaggi dopo il messaggio m_j, allora abbiamo che $C_A = d_G(t_j, s) + C_D$ e $C_{A'} = d_G(t_i, s) + C_D$, dove C_D è il costo per le restanti azioni, e dunque è uguale in entrambi gli schedule. In questo caso quindi abbiamo che il risparmio totale è pari a $COST(S|_{a_j}) - COST(S'|_{a_j}) = 0$. Se invece l'agente a_j non consegna altri

messaggi in S dopo aver consegnato il messaggio m_j, allora abbiamo che $C_A = 0$ e $C_{A'} = 0$, e dunque il secondo schedule, quello che consegna prima il messaggio m_i, ha un risparmio di

$$COST(S|_{a_j}) - COST(S'|_{a_j}) = d_G(s, t_i) - d_G(s, t_j) \geq 0$$

□

Possiamo adesso procedere con la dimostrazione finale.

Proprietà 4.1.4. Sia S uno schedule ristretto ottimo in forma canonica che consegna tutti i messaggi nelle rispettive destinazioni, e sia S_A lo schedule ristretto ritornato dall'algoritmo greedy. Supponiamo che S e S_A eseguono le stesse identiche azioni nello stesso identico ordine per la consegna dei primi k messaggi. Allora esiste uno schedule S^* tale che

1. S^* è feasible ed è in forma canonica, ovvero consegna tutti i messaggi in modo valido.
2. $COST(S^*) \leq COST(S)$, ovvero il costo dello schedule S^* non è peggiore del costo dello schedule S.
3. S^* e S_A eseguono le stesse identiche azioni nello stesso identico ordine per la consegna dei primi $k + 1$ messaggi.

Dimostrazione. Iniziamo ordinando i messaggi $m_{i_1}, m_{i_2}, ..., m_{i_m}$, in base alla distanza necessaria per la loro consegna: m_{i_1} è il messaggio la cui destinazione è la più vicina da s e m_{i_m} è il messaggio la cui destinazione è la più lontana da s.

Dato che lo schedule S_A segue la regola greedy, i primi k messaggi consegnati da S_A sono, in ordine, $m_{i_1}, m_{i_2}, ..., m_{i_k}$. Dalle ipotesi sappiamo anche che lo schedule S e lo schedule S_A eseguono le stesse identiche azioni nello stesso identico ordine per la consegna dei primi k messaggi. Segue che S e S_A potrebbero differire solamente a partire dall'azione successiva alla consegna del messaggio m_{i_k}. Andiamo ora a elencare tutti i possibili casi in cui lo schedule S può ricadere:

Dopo aver consegnato m_{i_k}, lo schedule S consegna $m_{i_{k+1}}$. Questo caso si spezza in due sottocasi, a seconda di quale agente S utilizza per la consegna. Se S utilizza lo stesso agente utilizzato da S_A per la consegna di $m_{i_{k+1}}$, allora basta porre $S^* := S$.

Se invece S utilizza un agente diverso da quello utilizzato da S_A, ovvero se S utilizza l'agente a_i, mentre S_A utilizza l'agente a_j per la consegna del messaggio $m_{i_{k+1}}$, con $i \neq j$, allora possiamo definire un nuovo schedule, lo schedule S^*, nel seguente modo: S^* è uguale allo schedule S, ma in S^* le azioni effettuate dall'agente a_i vengono scambiate con le azioni effettuate dall'agente a_j (e viceversa), a partire dalle azioni per la consegna del messaggio $m_{i_{k+1}}$.

Notiamo che S^* è uno schedule feasible il cui costo non è peggiore di S. Infatti, se S utilizza solamente l'agente a_i a partire dalla consegna del messaggio $m_{i_{k+1}}$, allora S^* utilizzerà solamente l'agente a_j a partire dalla consegna del messaggio $m_{i_{k+1}}$. Inoltre, dato che a_j è stato scelto da S_A, e dato che S_A è stato costruito tramite la regola greedy,

abbiamo che dopo la consegna di m_{i_k}, $d_G(a_j, s) \leq d_G(a_i, s)$. Infine, dato che i pesi sono uniformi, entrambi gli schedule pagheranno lo stesso costo per la restante traiettoria da percorrere. Ma allora $COST(S^*) \leq COST(S)$, e abbiamo fatto.

Nel caso in cui S utilizza entrambi gli agenti a_i e a_j dopo la consegna del messaggio $m_{i_{k+1}}$, allora abbiamo che $COST(S) = COST(S^*)$. Infatti, il costo di S^* potrebbe differire dal costo di S solamente per le azioni compiute dagli agenti a_i e a_j. Dato che però a_i e a_j vengono entrambi utilizzati da S, allora saranno anche entrambi utilizzati da S^*, e dunque entrambi gli agenti ad un certo punto dovranno essere spostati alla sorgente s sia nello schedule S che nello schedule S^*. Una volta che abbiamo entrambi gli agenti in s in entrambi gli schedule, dato che i pesi sono uniformi, non importa a quale agente si assegna quale traiettoria. Quindi, in questo caso, il costo per portare inizialmente gli agenti alla sorgente s dopo la consegna di m_{i_k} è lo stesso in entrambi gli schedule, e anche il costo delle restanti azioni è lo stesso. Ma allora $COST(S) = COST(S^*)$. Con questo abbiamo concluso il caso in cui lo schedule S consegna $m_{i_{k+1}}$ dopo aver consegnato m_{i_k}.

Dopo aver consegnato m_{i_k}, lo schedule S consegna m_{i_h}, con $h > k+1$. In questo caso abbiamo che il messaggio $m_{i_{k+1}}$ viene consegnato da S più tardi. Questo caso può essere diviso in due sottocasi, a seconda se la consegna di $m_{i_{k+1}}$ in S viene effettuata da un agente che è stato utilizzato tra la consegna di m_{i_k} e la consegna di $m_{i_{k+1}}$.

Se la consegna del messaggio $m_{i_{k+1}}$ viene effettuata da un agente a_j non utilizzato in S tra la consegna di m_{i_k} e la consegna di $m_{i_{k+1}}$, ci basta spostare la coppia di azioni di pickup e dropoff $(a_j, s, m_{i_k}, +)$, $(a_j, t_{i_k}, m_{i_k}, -)$, subito dopo l'azione di consegna del messaggio m_{i_k} per ottenere un nuovo schedule S^*. Notiamo che lo schedule S^* è feasible ed ha lo stesso costo dello schedule S. Infatti, dato che l'agente a_j non viene utilizzato in S tra la consegna di m_{i_k} e la consegna di $m_{i_{k+1}}$, il percorso che l'agente a_j aveva in S resta invariato in S^*, e dunque $COST(S^*) = COST(S)$. Così facendo ci siamo ricondotti ad un caso precedentemente dimostrato.

Infine, vediamo il caso in cui la consegna del messaggio $m_{i_{k+1}}$ viene effettuata da un agente a_j precedentemente utilizzato tra la consegna di m_{i_k} e la consegna di $m_{i_{k+1}}$. Notiamo che dall'ordinamento dei messaggi, abbiamo che l'agente a_j deve necessariamente consegnare un messaggio m_{i_l}, con $l > k+1$, e dunque con $d_G(s, t_{i_l}) \geq d_G(s, t_{i_{k+1}})$, tra la consegna di m_{i_k} (che può essere effettuata da agente diverso da a_j), e la consegna di $m_{i_{k+1}}$ (che è effettuata da a_j). Tramite l'utilizzo di una proprietà precedentemente dimostrata (proprietà 4.3), a partire dallo schedule S possiamo sempre ottenere uno schedule S', non peggiorando il costo, in cui gli agenti consegnano i messaggi rispetto alla distanza tra la destinazione e la sorgente. Per costruzione quindi, in S', l'agente a_j non può essere utilizzato tra la consegna del messaggio m_{i_k} e la consegna del messaggio $m_{i_{k+1}}$, in quanto altrimenti tale agente non rispetterebbe la proprietà che è stata imposta a S'. Così facendo ci siamo ricondotti ad un caso precedentemente dimostrato. □

4.2 Un algoritmo 2-approssimante per il caso di agenti non uniformi

Infine, andiamo ad analizzare le istanze che rispettano solo la restrizione **SMS**, ovvero le istanze in cui i pesi degli agenti non sono uniformi.

4.2.1 Algoritmo

Per quanto riguarda il caso in cui i pesi degli agenti non sono uniformi, è possibile definire un semplice algoritmo che fa consegnare tutti i messaggi ad un singolo agente. L'algoritmo procede nel seguente modo: per ogni agente a_i, considera la soluzione in cui tutti i messaggi sono consegnati da a_i in un ordine arbitrario. Notiamo, al fine dell'analisi, che tale soluziona ha costo al più

$$w_i \cdot [d_G(p_i, s) + 2 \cdot \sum_{i=1}^{m} d_G(s, t_i)]$$

Fra tutte le k soluzioni considerate, l'algoritmo restituisce la migliore, ovvero quella che minimizza il costo totale.

4.2.2 Analisi algoritmo

Questo semplice algoritmo riesce ad ottenere una $2-$approssimazione del problema. Per andare a vedere il perché, andiamo prima ad ottenere un lower bound sul costo di uno schedule ottimo.

Sia S_{OPT} uno schedule ottimo. Indichiamo con a_{min} l'agente di peso minimo utilizzato in S_{OPT}. Supponiamo poi, per semplificare l'analisi, che tutti gli agenti utilizzati in S_{OPT} hanno il peso pari a w_{min}. Osserviamo che questa scelta non può far altro che diminuire il costo dello schedule, e quindi un lower bound ottenuto con il peso degli agenti modificato in questo modo vale anche quando il peso degli agenti non viene toccato. Ora, dato che lo schedule deve essere feasible, ogni messaggio deve essere consegnato correttamente. Dunque, globalmente, i vari agenti dovranno percorrere, per consegnare il messaggio m_i, una distanza di almeno $d_G(s, t_i)$, e questo vale per ogni $i = 1, ..., m$. Ma allora tale schedule deve almeno costare $w_{min} \cdot \sum_{i=1}^{m} d_G(s, t_i)$.

Inoltre, al fine di consegnare i messaggi, almeno un agente si dovrà spostare dalla sua posizione iniziale alla sorgente s. Se l'agente a_{min} nello schedule S_{OPT} viene spostato dalla sua posizione iniziale p_{min} alla sorgente s, abbiamo sicuramente che lo schedule dovrà pagare un costo aggiuntivo di $w_{min} \cdot d_G(p_{min}, s)$. Se invece a_{min} non viene mai spostato, nello schedule S_{OPT}, dal nodo p_{min} alla sorgente s, allora vuol dire che a_{min} viene spostato in un nodo intermedio u al fine di aiutare nella consegna di un messaggio, diciamo il messaggio m_i. Notiamo che questo significa che un altro agente, diciamo l'agente a_j, è andato nel nodo s per raccogliere il messaggio m_i, e poi ha lasciato il messaggio m_i nel nodo u. Ma allora lo schedule S_{OPT} deve sicuramente pagare un costo aggiuntivo di $w_{min} \cdot [d_G(p_{min}, u) + d_G(s, u)]$, e dato che $d_G(p_{min}, u) + d_G(u, s) \geq d_G(p_{min}, s)$, abbiamo

che anche in questo caso lo schedule deve pagare un costo aggiuntivo di $w_{min} \cdot d_G(p_{min}, s)$. Dunque, in ogni caso, vale il seguente lower bound

$$COST(S_{OPT}) \geq w_{min} \cdot [d_G(p_{min}, s) + \sum_{i=1}^{m} d_G(s, t_i)]$$

Per finire, ci basta notare che lo schedule S ottenuto utilizzando la regola descritta in precedenza ha un costo tale che

$$\begin{aligned} COST(S) &\leq w_{min} \cdot [d_G(p_{min}, s) + 2 \cdot \sum_{i=1}^{m} d_G(s, t_i)] \\ &\leq w_{min} \cdot [2 \cdot d_G(p_{min}, s) + 2 \cdot \sum_{i=1}^{m} d_G(s, t_i)] \\ &= 2 \cdot COST(S_{OPT}) \end{aligned}$$

e dunque è una $2-$approssimazione dello schedule ottimo.

Capitolo 5

Conclusioni e problemi aperti

In questa tesi abbiamo studiato il problema del WeightedDeliver nel caso particolare in cui tutti i messaggi partono da un singolo nodo sorgente. Per questo caso abbiamo progettato due algoritmi polinomiali che implementano entrambi una strategia greedy. Il primo è un algoritmo esatto quando i pesi degli agenti sono uniformi. Il secondo algoritmo invece ottiene una 2-approssimazione anche quando i pesi degli agenti non sono uniformi.

Il nostro studio lascia ancora aperti diversi problemi. Ad esempio, bisogna ancora stabilire se il problema del WeightedDelivery sotto la restrizione **SMS** rimane un problema NP-hard. Un altro problema interessante riguarda il costo della collaborazione per le istanze single-source: sappiamo che per istanze generali non collaborare induce un costo di 2 nella soluzione trovata, però non sappiamo se per le istanze single-source il costo di non collaborare è più basso.

Bibliografia

[1] A. Bärtschi, J. Chalopin, S. Das, Y. Disser, D. Graf, J. Hackfeld, and P. Penna. Energy-efficient delivery by heterogeneous mobile agents. *CoRR*, abs/1610.02361, 2016.

[2] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962.

[3] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[4] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9:11–12, 1962.