Quickest Flow Over Time

↗ Literature Reviews	Literature Review [Paper Title]

FleischerSkutella.pdf

Problema del massimo flusso

Si vuole distribuire un certo "prodotto" da uno (o più) punti di produzione ad uno (o più) punti di utilizzo.

Dato un grafo orientato $G(V, A)$ in cui a ciascun arco $(i, j) \in A$ è associata una capacità massima u_{ij}

Dati due nodi distinti s e t dell'insieme V detti sorgente e pozzo (o origine e destinazione)

> Il problema del massimo flusso consiste nel determinare la massima quantità di flusso che è possibile inviare alla sorgente s alla destinazione t attraverso il grafo $G(V, A)$

- indichiamo sull'arco in nero le capacità degli archi
- indichiamo fra parentesi il flusso effettivo

Questo problema prevede più sorgenti e più destinazioni ma aggiungendo un nodo sorgente fittizio s^* e un nodo destinazione fittizio t^* , entrambi con un insieme di archi (s^*, s_i) e (t_i, t^*) a capacità illimitata, possiamo ricondurci al caso di una sorgente una destinazione.

Formulazione del problema

- **Variabili decisionali**

  $x_{ij} \geq 0$ quantità di flusso che attraversa l'arco $(i, j) \in A$

  $f \geq 0$ flusso totale inviato dalla sorgente alla destinazione

- **Funzione obiettivo**

  $max \;\; f$

- **Vincoli del problema**

  Continuità del flusso nei nodi intermedi, dal nodo sorgente deve uscire tutto il flusso inviato alla destinazione e nel nodo destinazione deve arrivare tutto il flusso inviato dall'origine. Come Kirchoff ai nodi : ***flusso entrante = flusso uscente.***

$$\sum_{e \in \delta^+(v)} x_{ij} - \sum_{e \in \delta^-(v)} x_{ij} = \begin{cases} f, & se \quad i = s \\ 0, & se \;\; \forall v \in V - \{s, t\} \\ -f & se \quad i = t \end{cases}$$

  Vincoli di capacità degli archi

$$0 \leq x_{ij} \leq u_{ij} \quad (i, j) \in A$$

Una generica soluzione ammissibile $\bar{x}$ rappresenta un flusso ammissibile sul grafo

Obiettivo del paper: fornire un algoritmo semplice e veloce per il problema generale del flusso più rapido (*Quickest Flow Over Time)* che è **NP-Hard.**

Poichè non si può trovare la soluzione esatta in tempo ragionevole, ci si accontenta di una soluzione **approssimata** con una garanzia di qualità.

L'algoritmo che sviluppano garantisce di trovare un piano di evacuazione il cui **tempo non è peggiore del doppio del tempo ottimale teorico (garanzia di 2-approssimazione).**

Il loro ragionamento si sviluppa in 3 passaggi logici:

1. Ponte tra Flusso Dinamico e Flusso Statico

Idea: creare un collegamento tra il mondo complesso dei flussi nel tempo (dinamici) e quello più semplice dei flussi classici (statici)

- **Da Dinamico a Statico:** Gli autori osservano che qualsiasi piano di evacuazione (un flusso dinamico f) che si completa entro un tempo T può essere convertito in un flusso statico x.

  > Immagina di "**fotografare**" il flusso totale passato su ogni strada durante l'intera evacuazione: questo è il tuo flusso statico.

  Questo flusso statico x ha una proprietà fondamentale: è **T-length-bounded** (a lunghezza limitata da T). Questo significa che il flusso x può essere scomposto in un insieme di percorsi, e ogni percorso P effettivamente utilizzato per trasportare persone ha una *"**lunghezza**" (somma dei tempi di transito τ_e lungo il percorso) **che non supera** T.*

  > Questo è intuitivo: se l'evacuazione è finita in tempo T, nessuno può aver preso un percorso che richiedeva più di T per essere completato

- **Da Statico a Dinamico:** viceversa, se hai un flusso statico x che è *T-lenght-bounded,* puoi trasformarlo in un piano di evacuazione (flusso dinamico g).

1.1 Algoritmo di Ford Fulkerson

Posto $x = 0$, e sia il grafo residuo $G_x(V, A(x)) = G(V, A)$

Finché esiste un **cammino aumentante** C sul grafo residuo $G_x(V, A(x))$

calcola $\delta = min_{(i,j)\in C}\ \ r_{ij}$

Per ogni arco $(i, j) \in A$

if $(i, j) \in C$

then $\ x_{ij} = x_{ij} + \delta$

if $(j, i) \in C$

then $\ x_{ij} = x_{ij} + \delta$

Aggiorna il grafo residuo $G_x = (V, A(x))$

- L'algoritmo termina quando sul grafo residuo non esistono più percorsi da s a t

1.2 Quickest Flow

Il problema del flusso più rapido (quickest flow) è strettamente correlato al problema di **calcolare un flusso s-t massimo nel tempo**. Il problema del flusso s-t più rapido richiede di inviare una quantità di flusso data da una sorgente s a un pozzo t nel minor tempo possibile.

Questo problema può essere risolto in tempo polinomiale integrando l'algoritmo di Ford e Fulkerson in un contesto di ricerca binaria. Inoltre, Burkard, Dlaska e Klinz hanno presentato un algoritmo più veloce che risolve il problema del flusso s-t più rapido in tempo fortemente polinomiale, utilizzando il metodo di ricerca parametrica di Megiddo.

Il problema del flusso più rapido può essere ulteriormente generalizzato per includere reti con costi sugli archi e problemi con più sorgenti e pozzi (***problema di trasbordo più rapido***), nonché il caso di più merci (***multicommodity flows***).

Ford e Fulkerson hanno mostrato come calcolare un flusso s-t massimo nel tempo riducendo questo problema a un problema di flusso a costo minimo statico. La soluzione ottimale di questo problema di flusso a costo minimo può essere trasformata in un flusso s-t massimale nel tempo scomponendola in flussi su percorsi. Un tale flusso è definito come "temporally repeated" e invia flusso a una certa velocità in ciascun percorso per un intervallo di tempo specifico.

Tuttavia, il problema del trasbordo più rapido, che coinvolge più sorgenti e pozzi per una singola merce, non è equivalente a un problema di flusso s-t massimo nel tempo per le reti statiche. A differenza dei problemi di flusso statico tradizionali, in cui il problema con molteplici sorgenti e pozzi può spesso essere ricondotto a un problema s-t singolo, questo non vale per i flussi nel tempo.

Hoppe e Tardos hanno affrontato questa complessità introducendo il concetto di "flussi decomponibili in catene" (chain decomposable flows). Questi flussi rappresentano una generalizzazione dei "flussi ripetuti temporalmente" (temporally repeated flows), consentendo di essere codificati in modo compatto come una raccolta di percorsi. Una caratteristica distintiva dei flussi decomponibili in catene è che i percorsi possono includere anche archi all'indietro (backward arcs), il che richiede un'analisi attenta per garantirne la fattibilità.

L'algoritmo proposto da Hoppe e Tardos è stato il primo a risolvere questo problema in tempo polinomiale. Nonostante questa pietra miliare, il loro algoritmo non è considerato pratico per applicazioni generali, poiché il suo tempo di esecuzione è di ordine polinomiale elevato e richiede l'uso di un oracolo.

2. Preliminaries

Consideriamo problemi di routing su una rete $\mathcal{N} = (V, A)$ con $n := |V|$ nodi e $m := |A|$ archi. Ogni arco $e \in A$ ha associato:

- un ***transit time*** integrale o lunghezza τ_e
- una capacità u_e
- un coefficiente di costo c_e che vale per unità di flusso che attraversa e

Un arco dal nodo v al nodo w è indicato anche come (v, w) e in questo caso diremo che $head(e) = w$ e $tail(e) = v$.

2.1 Flussi statici

Iniziamo con una definizione di flussi **single commodity.**

Un set di terminali $S \subseteq V$ è diviso in sorgenti(sources) S^+ e pozzi (sink) S^-.

- Ogni ***nodo sorgente*** $v \in S^+$ ha un'offerta

  $D_v \geq 0 \;\; \forall v \in S^+$
- Ogni ***nodo destinazione*** $v \in S^-$ ha una domanda pari a

  $D_v \leq 0 \;\; \forall v \in S^-$

Quindi complessivamente $\sum_{v \in S} D_v = 0$.

Un flusso statico x su una rete $\mathcal{N}$ assegna ad ogni arco e un flusso non negativo di valore x_e tale che:

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} = 0 \qquad \forall v \in \{V\} - S$$

In particolare il flusso statico x soddisfa domanda e offerta se

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} = D_v \qquad \forall v \in S$$

> Continuità del flusso nei nodi intermedi, dal nodo sorgente deve uscire tutto il flusso inviato alla destinazione e nel nodo destinazione deve arrivare tutto il flusso inviato dall'origine. Come Kirchoff ai nodi : ***flusso entrante = flusso uscente.***

Diremo che un flow è ***feasible*** se rispetta i vincoli di capacità $x_e \leq u_e \quad \forall e \in A$

Definiamo costo di un flusso statico x come

$$c(x) := \sum_{e \in A} c_e x_e$$

Passiamo ai flussi **multicommodity**

La rete gestisce più "commodity", rappresentate da un insieme $K = \{1, \ldots, k\}$

- Ogni merce $i \in K$ ha il proprio insieme di terminali $S_i = S_i^+ \cup S_i^- \subseteq V$ *e le proprie offerte/domande* $D_{v,i}$
- *Un flusso multicommodity statico* x *su* $\mathcal{N}$ *assegna un valore di flusso* ***non negativo*** *di valore* $x_e^i : x^i := (x_e^i)_{e \in A}$ *per ogni* ***coppia arco-merce*** (e, i)
- *Per ogni commodity* i*, il vettore di flussi* $x^i := (x_e^i)_{e \in A}$ deve comportarsi come un flusso a singola commodity, rispettando i propri vincoli di conservazione e domanda/offerta

Il vincolo di capacità per un arco e si applica al flusso totale che lo attraversa, ovvero la somma dei flussi di tutte le merci: $x_e := \sum_{i \in K} x_e^i \leq u_e \quad \forall e \in A$.

Il costo di un flusso multicommodity x è la somma dei costi di tutte le merci su tutti gli archi:

$$c(x) = \sum_{e \in A} \sum_{i \in K} c_{e,i} x_e^i$$

2.2 Flussi nel tempo (opzionale)

Un "flusso nel tempo" (flow over time) è definito su una rete con un orizzonte temporale T. È rappresentato da una collezione di funzioni misurabili secondo Lebesgue, $f_{e,i} : [0, T) \to R^+$, dove $f_{e,i}(\theta)$ indica il tasso di flusso (per unità di tempo) della merce `i` che entra nell'arco `e` al tempo θ. I tempi di transito sono fissi, il che significa che il flusso attraversa ciascun arco a una velocità uniforme. Di conseguenza, il flusso $f_{e,i}(\theta)$ di merce i che entra nell'arco e al tempo θ arriva alla testa dell'arco e al tempo $\theta + \tau_e$. Per rispettare l'orizzonte temporale T, è richiesto che $f_{e,i}(\theta)$ per $\theta \in [T - \tau_e, T)$.

Un flusso nel tempo f è considerato **ammissibile** (feasible) se rispetta le capacità degli archi e le condizioni di conservazione del flusso.

Le sue proprietà principali, con particolare attenzione all'ammissibilità, sono le seguenti:

- **Vincoli di Capacità**: La capacità u_e di un arco e funge da limite superiore per il tasso di flusso che può entrare nell'arco per unità di tempo. Pertanto, la capacità è interpretata come una capacità per unità di tempo, e i vincoli richiedono che $f_e(\theta) \leq u_e$ per ogni $\theta \in [0, T)$ e per ogni arco $e \in A$, dove $f_e(\theta)$ è il flusso totale che entra nell'arco e al tempo θ.
- **Conservazione del Flusso**: Esistono due modelli distinti per la conservazione del flusso:
  - **Con Stoccaggio in Nodi Intermedi**: Questo modello permette di immagazzinare il flusso (inventario) in nodi che non sono né sorgenti né pozzi, prima di proseguire il trasporto. I vincoli di conservazione del flusso sono integrati nel tempo per prevenire un deficit in qualsiasi nodo:

$$\int_0^{\xi} \Big(\sum_{e \in \delta+(v)} f_{e,i}(\theta) - \sum_{e \in \delta-(v)} f_{e,i}(\theta - \tau_e) \Big) d\theta \leq 0$$
$$\forall i \in K, \xi \in [0, T), v \in V - S_i^+$$

    Inoltre, è richiesto che l'uguaglianza si verifichi per $i \in K, \xi = T$, e $v \in V - S_i$, assicurando che nessun flusso rimanga nella rete dopo il tempo T

  - **Senza Stoccaggio in Nodi Intermedi**: In questo modello, la condizione di uguaglianza nella conservazione del flusso

$$\int_0^{\xi}(\sum_{e\in\delta+(v)} f_{e,i}(\theta) - \sum_{e\in\delta-(v)} f_{e,i}(\theta-\tau_e))d\theta \leq 0$$
$$\forall i \in K, \forall \xi \in [0,T), v \in V - S_i$$

Un flusso nel tempo f soddisfa le forniture e le richieste se, entro il tempo T, il flusso netto in uscita da ciascun nodo terminale $v \in S_i$ della merce i è uguale alla sua fornitura $D_{v,i}$. Questo è formalmente espresso come:

$$\int_0^{T}(\sum_{e\in\delta+(v)} f_{e,i}(\theta) - \sum_{e\in\delta-(v)} f_{e,i}(\theta-\tau e))d\theta = D_{v,i}$$
$$\forall i \in K, \forall v \in S_i.$$

2.3 Massimo flusso nel tempo

Ford e Fulkerson hanno dimostrato che questo problema può essere risolto in modo efficiente riducendolo a un problema di **flusso a costo minimo statico** sulla rete data.

L'obiettivo di questo problema di flusso a costo minimo è massimizzare il flusso che arriva al pozzo entro il tempo T. La funzione obiettivo utilizzata è la seguente:

$$max \quad T|x| - \sum_e \tau_e x_e$$

Dove:

- T: rappresenta l'orizzonte temporale dato, ovvero il tempo massimo disponibile per il completamento del flusso.
- $|x|$: indica il valore totale del flusso statico x.
- τ_e: è il tempo di transito o la lunghezza dell'arco e.
- x_e: rappresenta il valore del flusso statico sull'arco e.

Questa formulazione consente di convertire una soluzione ottimale x a questo problema di flusso a costo minimo in un flusso massimo nel tempo. Un flusso statico x può essere scomposto in una somma di flussi x_p su percorsi semplici P. Il flusso nel tempo risultante, chiamato ***"flusso ripetuto temporalmente" (temporally repeated flow***), invia il flusso a una velocità x_P in ogni percorso P durante l'intervallo di tempo $[0, T - \tau(P))$, dove $\tau(P)$ è la somma dei tempi di transito degli archi sul percorso P.

Il valore di questo flusso f è calcolato come:

$$|f| = \sum_{P\in P}(T - \tau(P))x_P = T|x| - \sum_e \tau_e x_e$$

Dove:

- $|f|$: è il valore del flusso nel tempo.
- P: è l'insieme di tutti i percorsi semplici $s-t$.
- $\tau(P) = \sum_{e\in P} \tau_e$: è la somma dei tempi di transito degli archi sul percorso P.
- x_P: è il flusso inviato lungo il percorso P.
- La seconda uguaglianza ($= T|x| - \sum_e \tau_e x_e$) deriva dal fatto che $(x_P)P \in \mathcal{P}$ è una scomposizione in percorsi di x.

La fattibilità di f deriva direttamente dalla fattibilità di x.

Flussi Più Rapidi (Quickest Flows)

Un problema strettamente correlato ai ***flussi massimi s-t*** nel tempo è il problema del **flusso più rapido s-t**. In questo caso, l'obiettivo è inviare una quantità di flusso d data dalla sorgente s al pozzo t nel minor tempo possibile (orizzonte temporale minimo T).

Questo problema può essere risolto in tempo polinomiale integrando l'algoritmo di Ford e Fulkerson in un framework di ricerca binaria.

→ il nostro approccio.

Questo problema può essere generalizzato al **problema del trasbordo più rapido** (*quickest transshipment proble*m), che considera un vettore di forniture e richieste ai nodi e mira a trovare un flusso nel tempo che soddisfi tutte le forniture e richieste nel tempo minimo. Una ulteriore generalizzazione è il caso di **più merci** (multicommodity flows).

> Il problema più generale considerato è il **problema di trasbordo multicommodity più rapido con costo limitato**, definito come segue:
>
> **Input:** Una rete (grafo orientato) con capacità, costi e tempi di transito sugli archi; k merci, ciascuna specificata da un insieme di sorgenti e pozzi con forniture e richieste; e un budget di costo C.
>
> **Compito:** Trovare un flusso multicomodità nel tempo che soddisfi tutte le forniture e richieste con costo al massimo C e con orizzonte temporale T minimo.

Questo problema è generalmente **NP-hard**, anche per casi semplici come il flusso a costo minimo nel tempo o i flussi multicommodity nel tempo. Di conseguenza, la ricerca si concentra spesso su algoritmi di approssimazione, come gli "schemi di approssimazione completamente polinomiale" (***Fully Polynomial-Time Approximation Schemes - FPTAS***), che permettono di trovare soluzioni vicine all'ottimo in tempo polinomiale.

È importante notare che i limiti di tempo sono talvolta espressi in termini di T^*, l'orizzonte temporale ottimale. Poiché capacità e tempi di transito sono interi, si può assumere un limite superiore grossolano per T^* dato da $T^* \leq \sum_i d_i + \sum_e \tau_e$, dove d_i sono le domande e τ_e i tempi di transito. ***Finché la dipendenza da T^* è polilogaritmica, il limite risultante è polinomiale nella dimensione dell'input.***

3. A simple two-approximation algorithm

3.1. Length-bounded static flows.

I length-bounded static flows rappresentano una generalizzazione dei flussi statici standard che incorporano l'elemento temporale dei tempi di transito. L'obiettivo è definire flussi che utilizzino ***percorsi "ragionevoli" in termini di tempo di transito.***

Un flusso statico (multicommodity) x è definito ***T-length-bounded*** se il flusso di ogni merce $i \in K$ può essere scomposto in una somma di flussi x^i_P su cammini semplici $P \in \mathcal{P}_i$ (dove $\mathcal{P}_i$ è l'insieme di tutti i cammini dalla sorgente di i a uno dei suoi pozzi), tali che la lunghezza $\tau(P)$ di qualsiasi cammino $P \in \mathcal{P}_i$ con $x_P > 0$ sia al massimo T.

- La lunghezza $\tau(P)$ di un cammino P è la somma dei tempi di transito (τ_e) degli archi che lo compongono.

Questo problema può essere formulato come un PLI per trovare un flusso a costo minimo con vincoli di lunghezza. Assumendo per semplicità che ogni merce $i \in K$ abbia una singola sorgente s_i e un singolo pozzo t_i con una domanda d_i, il problema di flusso a costo minimo con vincolo di lunghezza può essere scritto come segue:

Formulazione

Sia $P_T^i := \{P \in \mathcal{P}_i | \tau(P) \leq T\}$ l'insieme di tutti i cammini $s_i - t_i$ la cui lunghezza è limitata superiormente da T. Sia
$c_i(P) := \sum_{e \in P} c_{e,i}$ il costo del cammino P per la merce i, dove $c_{e,i}$ è il coefficiente di costo associato all'arco e e alla merce i.

Funzione Obiettivo: Minimizzare il costo totale.

$$min \sum_{i \in K} \sum_{P \in P_T^i} c_i(P) x_P^i$$

Vincoli:

1. **Domanda/Offerta**: Per ogni merce $i \in K$, la somma dei flussi su tutti i cammini $T-$limitati deve soddisfare o superare la domanda d_i.

$$\sum_{P \in P_T^i} x_P \geq d_i \ \ \forall i \in K$$

2. **Capacità degli archi**: Per ogni arco $e \in A$, la somma dei flussi di tutte le merci che lo utilizzano non deve superare la sua capacità u_e.

$$\sum_{i \in K} \sum_{P \in P_T^i : e \in P} x_P \leq u_e \quad \forall e \in A$$

3. **Non-negatività del flusso**: Il flusso su ogni cammino deve essere non negativo.

$$x_P \geq 0 \qquad \forall i \in K, P \in P_T^i$$

Questo problema è NP-Hard, anche per il caso di una singola merce, ma può essere approssimato con precisione arbitraria in tempo polinomiale.

Il duale di questo programma lineare è il seguente:

$$max \sum_{i \in K} d_i z_i - \sum_{e \in A} u_e y_e$$

soggetto a:

$$\sum_{e \in P} (y_e + c_{e,i}) \geq z_i$$
$$\forall i \in K, P \in P_T^i z_i, y_e \geq 0$$
$$\forall i \in K, e \in A$$

Il problema di separazione per questo duale è formulato come un problema di cammino minimo con vincolo di lunghezza.

3.2. The approximation algorithm.

L'algoritmo **LengthBounded** è un algoritmo di approssimazione proposto per il problema del **trasbordo multicomodità più rapido con costo limitato**. Il suo obiettivo è trovare un flusso nel tempo che soddisfi tutte le forniture e richieste con un costo limitato C e un orizzonte temporale (***makespan***) il più breve possibile.

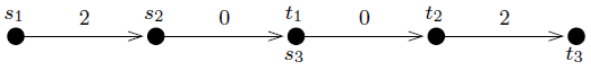


FIG. 2. *An instance of the quickest multicommodity flow problem containing three commodities $i = 1, 2, 3$, each with a single source s_i and a single sink t_i. Commodities 1 and 3 have demand value 1; commodity 2 has demand value 2. The numbers at the arcs indicate the transit times; all arcs have unit capacity. Note that an arc with 0 transit time and capacity 1 takes no additional time to cross but lets only one unit of flow through per unit time. A quickest flow with waiting at intermediate nodes allowed takes three time units and stores one unit of commodity 2 at the intermediate node $t_1 = s_3$ for two time units. However, if flow cannot be stored at intermediate nodes, an optimal solution takes time 4.*

Input dell'Algoritmo:

- Un'istanza del problema di trasbordo multicommodity più rapido con un budget di costo C.
- Un orizzonte temporale T (tentativo).
- Un valore di precisione $\varepsilon > 0$.

Output dell'Algoritmo:

- Un flusso ammissibile nel tempo che soddisfa tutte le forniture e richieste, con un makespan massimo di $(2 + \varepsilon)T$ e un costo limitato da C.
- Oppure l'indicazione che T è strettamente inferiore al makespan ottimale (T^*).

Passaggi dell'Algoritmo:

1. **Calcolo di un flusso statico vincolato per lunghezza (T-length-bounded):**
   L'algoritmo cerca di calcolare un flusso statico
   `x` che soddisfi tre condizioni:
   - `x` **deve essere (1 +** `ε`**)T-length-bounded:** Questo significa che il flusso di ogni merce deve poter essere scomposto in percorsi semplici la cui lunghezza totale (somma dei tempi di transito sugli archi) sia al massimo `(1 + ε)T`. Se il problema di trovare un flusso statico T-length-bounded è NP-difficile, è possibile trovarne uno approssimato con precisione arbitraria in tempo polinomiale.
   - `x` **deve soddisfare una frazione** `1/T` **delle forniture e delle richieste:** Questa è una proprietà che deriva dall'aver "mediato" un flusso nel tempo su un intervallo `T`.
   - **Il costo** `c(x)` **deve essere al massimo** `C/T` **:** Anche questa proprietà è una conseguenza della relazione tra un flusso nel tempo e il suo flusso statico corrispondente.
   - Se non esiste un flusso statico che soddisfi queste condizioni, l'algoritmo si ferma e indica che il `T` fornito in input è troppo piccolo (`T < T*`).
2. **Conversione del flusso statico in un flusso nel tempo:**
   Se il flusso statico
   `x` viene trovato, esso viene convertito in un flusso nel tempo che soddisfa tutte le forniture e richieste con un makespan di al massimo `(2 + ε)T` e un costo limitato da `C`. Questa conversione si basa sul fatto che un flusso statico `x` (che è `T` length-bounded) può essere trasformato in un flusso nel tempo `g` che invia flusso su ciascun percorso `P` a una data velocità per `T` unità di tempo, attendendo poi per un massimo di `T` unità di tempo aggiuntive affinché tutto il flusso raggiunga la destinazione. Questo processo non richiede lo stoccaggio di flusso in nodi intermedi.

ALGORITHM LENGTHBOUNDED

Input: An instance of the quickest multicommodity transshipment problem with cost bound C; tentative time horizon T; precision $\varepsilon > 0$.

Output: A flow over time satisfying all supplies and demands with makespan at most $(2+\varepsilon)T$ and cost bounded by C; or the information that T is strictly smaller than the optimal makespan.

1. Compute a static flow x such that
   - x is $(1+\varepsilon)T$-length bounded;
   - x satisfies a fraction $1/T$ of the supplies and demands;
   - $c(x) \leq C/T$;

   or decide that no such flow exists. In the latter case stop and output "$T < T^*$".
2. Turn x into a flow over time satisfying all supplies and demands with makespan at most $(2+\varepsilon)T$ and cost bounded by C (see discussion in section 3.2).

FIG. 3. *The core of the $(2+\varepsilon)$-approximation algorithm.*

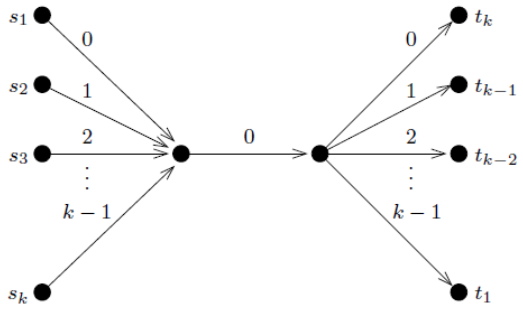


FIG. 4. *An instance with k commodities showing that the analysis in the proof of Theorem 3.3 is tight. All arcs have unit capacity and transit times as depicted above. The demand value of every commodity is 1. A quickest flow needs $T^* = k$ time units. However, any static flow can satisfy at most a fraction of $1/k$ of the demands. In particular, the makespan of the resulting flow over time is at least $2k-1$.*

Garanzia di Approssimazione:
Quando l'algoritmo LengthBounded viene integrato in un processo di ricerca binaria per determinare il makespan ottimale
`T*`, esso è in grado di trovare una soluzione con un makespan che è al massimo $(2+\varepsilon)$ volte il makespan ottimale `T*`, pur rispettando il budget di costo `C`. L'algoritmo produce soluzioni che non prevedono lo stoccaggio del flusso in nodi intermedi.

3.3. Avoiding the length-bounded flow computation

Il paragrafo 3.3 introduce un metodo per migliorare l'algoritmo di approssimazione (2+ε) presentato in precedenza, eliminando la necessità di eseguire un calcolo di flusso con limite di lunghezza, che può essere computazionalmente oneroso.

A differenza dei flussi ripetuti nel tempo di Ford Fulkerson, i flussi ottenuti dalla decomposizione di flussi statici con limite di lunghezza T (come descritto in 3.2) non sfruttano necessariamente i percorsi per la massima durata possibile (fino all'orizzonte temporale 2T), ma vengono interrotti al tempo T.

Per trasformare questi flussi in "flussi ripetuti temporalmente", l'intervallo di tempo$[0, T)$durante il quale il flusso viene inviato su ciascun percorso P viene esteso. L'obiettivo è fare in modo che l'ultimo flusso su un percorso P arrivi a destinazione esattamente al tempo $2T$. Il nuovo intervallo esteso diventa quindi $[0, 2T - \tau(P))$, dove $\tau(P)$ è il tempo di transito totale lungo il percorso P

Per compensare l'intervallo di tempo prolungato, la portata del flusso x_i^P su ciascun percorso P viene riscalata a

$$\tilde{x}_i^P = \frac{T}{(2T - \tau(P))} \cdot x_i^P \leq x_P^i$$

Questo riscalamento ***assicura che il flusso totale inviato su quel percorso rimanga invariato***, anche se distribuito su un intervallo di tempo più lungo, e che il flusso risultante rispetti le capacità degli archi poiché la portata del flusso su qualsiasi percorso non viene aumentata ($\tilde{x}_i^P \leq x_i^P$)

Per i problemi di flusso senza costi associati agli archi, questa osservazione permette di sostituire il calcolo di flusso con limite di lunghezza (spesso oneroso) con un calcolo di flusso standard (più rapido)

- I tempi di transito sugli archi vengono interpretati come coefficienti di costo

Il problema può essere formulato come un problema di ottimizzazione che ***minimizza il tempo T*** soggetto a vincoli che assicurano il soddisfacimento della domanda per ogni merce, tenendo conto dei tempi di transito e del flusso

Sebbene la formulazione sia non lineare, ***può essere risolta in tempo polinomial***e trasformandola in un problema di circolazione multicommodity parametrico

Teorema di approssimazione

> ***Theorem 3.4.*** There exists a 2-approximation algorithm for the quickest multicommodity transshipment problem. Moreover, the computed solution does not store flow at intermediate nodes.

Limitazione con l'Introduzione dei Costi:

- Il miglioramento (Teorema 3.4) non può essere generalizzato ai problemi che includono costi. Questo perché il costo di un flusso ripetuto temporalmente non è determinato unicamente dal flusso statico sottostante, ma dipende anche dalla specifica decomposizione del percorso scelta.

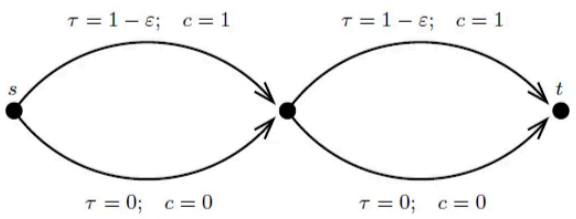


FIG. 5. *An instance showing that the cost of a temporally repeated flow depends on the particular path-decomposition of the underlying static s-t-flow. Consider the static flow of value 2 in the depicted network with flow value 1 on every arc. The time horizon is set to 2. There are two possible decompositions of this static flow into a sum of two path-flows. One of them leads to a temporally repeated flow of cost $2 + 2\varepsilon$; the other one has cost 2ε.*

- Trovare una decomposizione di percorso di un dato flusso statico che produca il flusso ripetuto temporalmente più economico è un problema NP-difficile, dimostrabile tramite una riduzione dal problema della Partizione

4. Schemi di approssimazione (non trattati nel nostro progetto)

4.1 Reti tempo espanse condensate

Le Reti Espanse nel Tempo Tradizionali

- Nel modello a tempo discreto, i flussi nel tempo possono essere descritti e calcolati in **reti espanse nel tempo**. Una rete espansa nel tempo, indicata come $\mathcal{N}^T$, è creata replicando il grafo statico originale $\mathcal{N} = (V, A)$ per ogni passo temporale discreto, da V_0 a V_{T-1}. Per ogni arco $e = (v, w)$ nel grafo statico e per ogni istante di tempo θ (dove $0 \leq \theta < T - \tau_e$), esiste un arco e_θ dalla copia v_θ alla copia $w_{\theta+\tau_e}$. Questi archi mantengono la capacità e il costo dell'arco originale.

- **Archi di holdover (stoccaggio)**: Vengono introdotti anche archi di "holdover" (o stoccaggio) con capacità infinita, da v^θ a $v^{\theta+1}$, per tutti i nodi e gli intervalli di tempo, modellando la possibilità di mantenere il flusso in un nodo intermedio.

Questo approccio trasforma un problema di flusso nel tempo in un problema di flusso statico classico sulla rete espansa nel tempo, che può essere risolto con algoritmi noti.

Il problema principale è che la **dimensione di $\mathcal{N}^T$ cresce linearmente con T (l'orizzonte temporale)**. Nel caso peggiore, T può essere esponenziale rispetto alla dimensione dell'input del problema, rendendo la soluzione proibitivamente costosa e gli algoritmi risultano "pseudopolinomiali".

Le Reti Espanse nel Tempo "Condensate"

Per superare il problema della dimensione esponenziale, il paper propone l'uso di **reti espanse nel tempo "condensate"**. L'idea è di utilizzare una **discretizzazione del tempo più grossolana**.

- Se tutti i tempi di transito degli archi sono multipli di un intervallo $\Delta > 0$, e se T/Δ è un intero, allora invece di una rete espansa a unità singola, si può utilizzare una rete condensata $N^{T/\Delta}$ che contiene solo (T/Δ) copie del set di nodi V. Ogni copia V^θ in $N^{T/\Delta}$ corrisponde al flusso attraverso V nell'intervallo $[\theta\Delta, (\theta + 1)\Delta)$.
- **Capacità Modificate**: In questa rete condensata, le capacità degli archi vengono **moltiplicate per** Δ per riflettere il fatto che ogni arco ora corrisponde a un intervallo di tempo di lunghezza Δ.

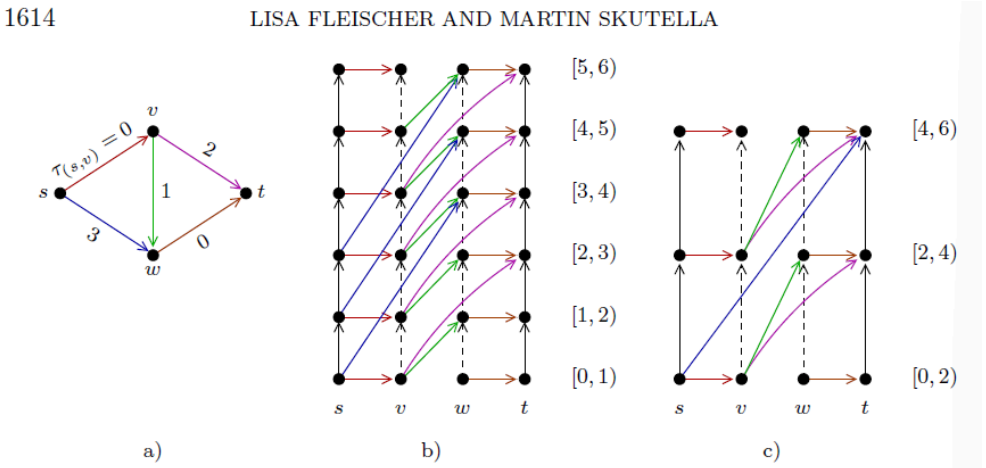


FIG. 6. (a) *A static network $\mathcal{N}$ with transit times on the arcs, one source s, and one sink t;* (b) *the corresponding time-expanded network $\mathcal{N}^T$ with time horizon $T = 6$;* (c) *the condensed time-expanded network $\mathcal{N}^T/\Delta$ with $\Delta = 2$. Notice that the transit time of arc (s, w) has been rounded up to 4 in $\mathcal{N}^T/\Delta$ since the original transit time 3 is not a multiple of Δ (see also section 4.2).*

Corrispondenza tra Flussi

Il cuore di questa sezione è la dimostrazione che i flussi sulla rete originale e sulla rete condensata sono equivalenti o strettamente correlati:

- **Lemma 4.1**: Se tutti i tempi di transito degli archi sono multipli di Δ e T/Δ è un intero, allora:
  - Qualsiasi flusso (multicommodity) nel tempo che si completa entro il tempo T nella rete originale corrisponde a un **flusso statico di uguale costo** in $N^{T/\Delta}$. Questo si ottiene mediando il valore del flusso in ogni intervallo di tempo $[\theta\Delta, (\theta + 1)\Delta)$.
  - Qualsiasi flusso statico in $N^{T/\Delta}$ corrisponde a un **flusso nel tempo di uguale costo** che si completa entro il tempo T nella rete originale. In questo caso, il flusso statico su un arco in $N^{T/\Delta}$ viene diviso per Δ e inviato per Δ unità di tempo.
  - Questa lemma sottolinea la **conservazione del valore e del costo** del flusso.

- **Corollario 4.2**: Se si abbandona la condizione che T/Δ sia un intero (cioè, il makespan T non è necessariamente un multiplo esatto di Δ), il risultato è leggermente più debole:
  - Un flusso nel tempo che si completa entro T corrisponde a un flusso statico di uguale valore e costo in $N^{T/\Delta}$.
  - Un flusso statico in $N^{T/\Delta}$ corrisponde a un flusso nel tempo di uguale valore che si completa **prima del tempo** $T + \Delta$. Questo introduce una leggera perdita nella precisione del makespan, ma la mantiene entro un margine controllabile.

4.2 Outline of an approximation scheme

- L'approccio si basa sulla "discretizzazione più grossolana del tempo" per ridurre la dimensione delle reti espanse nel tempo.
- Vengono arrotondati per eccesso i tempi di transito degli archi al multiplo più vicino di un opportuno intervallo di tempo Δ (o #).
- Si risolve il problema di flusso statico sulla corrispondente rete espansa nel tempo condensata (come discusso nella nostra precedente conversazione sulla Figura 6(c)).
- Infine, la soluzione ottenuta viene ritrasformata nel contesto dei tempi di transito originali

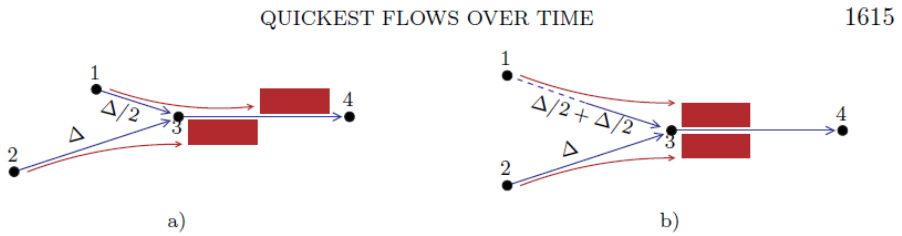
Due Condizioni Cruciali per Soluzioni di Buona Qualità

Per garantire che questo approccio porti a soluzioni con garanzie di approssimazione dimostrabili, devono essere soddisfatte due condizioni principali:

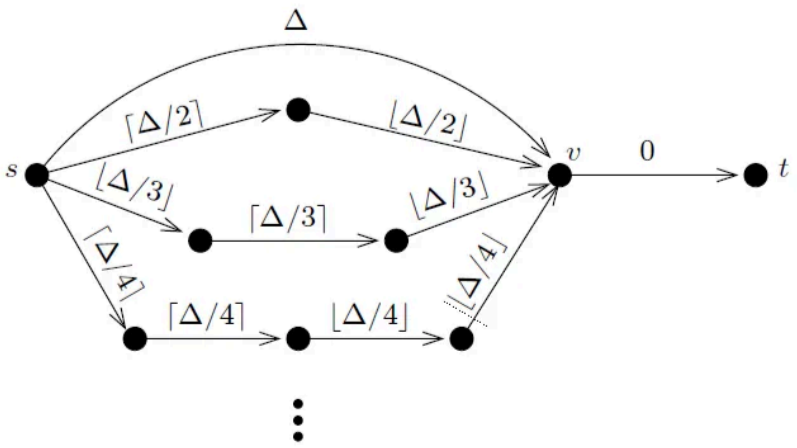
- **Condizione I:** Il makespan (tempo di completamento) di una soluzione ottimale per l'istanza con i tempi di transito aumentati (quella sulla rete condensata) deve approssimare il makespan di una soluzione ottimale nell'impostazione originale del problema
- **Condizione II**: La soluzione ottenuta dall'istanza con i tempi di transito aumentati deve essere trasformabile in un flusso nel tempo con le lunghezze degli archi originali senza una perdita eccessiva del valore del flusso

Gli autori presentano esempi per mostrare che queste condizioni non sono semplici da soddisfare e sollevano sfide significative:

- **Problema relativo alla Condizione II**



  Come discusso in precedenza, l'arrotondamento dei tempi di transito può portare a problemi di congestione. La Figura 7(a) mostra un flusso non congestionato nella rete originale. Tuttavia, se i tempi di transito sono arrotondati (Figura 7(b)), due pacchetti di flusso che prima usavano un arco in tempi distinti potrebbero arrivare contemporaneamente al nodo 3 e cercare di usare l'arco (3,4) nello stesso intervallo di tempo. Questo causa un "collo di bottiglia" o "congestione" sull'arco (3,4) che non esisteva prima, dimostrando che la trasformazione diretta può portare a una perdita di valore del flusso o a soluzioni infattibili
- **Un Altro Problema** (Figura 8)



La Figura 8 illustra un caso in cui, arrotondando i tempi di transito al multiplo più vicino di Δ, si potrebbe inviare un flusso simultaneo su più percorsi che, nella rete originale, avrebbero tempi di arrivo sfalsati sull'arco finale. Quando si tenta di interpretare questo flusso nella rete con i tempi di transito originali, ogni percorso potrebbe cercare di utilizzare un arco nella stessa finestra temporale, causando "un grande collo di bottiglia"

> If all transit times are rounded up to the nearest multiple of Δ, we may send Δ units of flow simultaneously on each path from s to t, and each path will use arc (v, t) in a distinct interval of time. If we try to interpret this flow in the network ***with original transittimes***, however, ***each path-flow will try to use arc (v, t) in the same time interva***l,causing a ***large bottleneck***.

Questo enfatizza ulteriormente la difficoltà di tradurre accuratamente i flussi dalla rete arrotondata a quella originale.

- ***Soluzione per la Condizione II:*** Permettere lo Stoccaggio
  Un modo per gestire il problema evidenziato dalla Condizione II è permettere lo stoccaggio del flusso nei nodi intermedi

L'***Osservazione 4.3*** afferma che se la lunghezza di un arco e viene aumentata di Δ' a causa dell'arrotondamento ($\tau'_e = \tau_e + \Delta'$), ciò può essere emulato nella rete originale introducendo un tempo di attesa di $\tau'_e - \tau_e$ unità alla testa dell'arco e. Questo significa che il flusso viene trattenuto nel nodo di arrivo per un certo periodo prima di essere inoltrato.

4.3 Grafi Aciclici con Stoccaggio

Nei grafi **senza cicli**, esiste un **ordinamento topologico** $\{v_0, v_1, ..., v_{n-1}\}$ dei nodi. Questo permette di gestire facilmente i ritardi nei flussi, perché il flusso **non può tornare indietro**.

La possibilità di stoccaggio è cruciale perché, come evidenziato dalla Figura 7 e discussione nel paragrafo 4.2, l'arrotondamento dei tempi di transito può creare problemi di congestione e colli di bottiglia che, in assenza di stoccaggio, porterebbero a una perdita di valore del flusso o all'infattibilità. ***Con lo stoccaggio, questo problema può essere gestito semplicemente "trattenendo" il flusso nei nodi per il tempo aggiuntivo causato dall'arrotondamento.***

Il problema specifico affrontato in questo contesto è quello del flusso multicomodity più rapido con costo limitato (quickest cost-bounded multicommodity flow)

Idea dell'Algoritmo (FPTAS):

1. Si fissa un tempo massimo TTT.
2. Si sceglie un parametro $\theta := \frac{\epsilon T}{n}$

3. Si **arrotondano verso l'alto i tempi di transito** degli archi alla **più vicina multipla di θ\thetaθ**.
4. Si costruisce il **grafo tempo-espanso condensato (θ-condensed)**.
5. Si calcola un flusso f in questo grafo statico.
6. Si **corregge** f per ottenere un nuovo flusso $\hat{f}$ che tiene conto dei ritardi introdotti dall'arrotondamento.

Per ogni arco $e = (v_i, v_j)$, si ritarda il flusso in uscita da v_i di $i\theta$ unità di tempo:

$\hat{f}_e(t) := f'_e(t - i\theta)$

Questo significa che il flusso attraversa lo stesso arco nello stesso modo, **solo spostato avanti nel tempo** di una quantità proporzionale all'indice del nodo di partenza.

Perché tutto questo funziona

- Il ritardo **massimo** accumulato lungo un cammino da v_0 a v_{n-1} sarà $(n-1)\theta = \epsilon T$
- Quindi il tempo totale richiesto sarà al massimo $T + \epsilon T = T(1+\epsilon)$

L'algoritmo proposto utilizza una strategia **iterativa**, tipicamente una **ricerca binaria** sul **makespan ottimale** T^*, per trovare la minima durata entro cui soddisfare le domande multicommodity entro un budget di costo C. Per ogni tentativo di makespan T, e per un parametro di precisione $\epsilon > 0$, il nucleo dell'algoritmo (chiamato **FPTAS-Core**) esegue i seguenti passi:

---

1. Scelta del passo temporale Δ e costruzione di un makespan aumentato

- Si fissa un intervallo temporale $\Delta := \frac{\epsilon T}{n}$, dove nn è il numero di nodi del grafo.

  Questa scelta serve a controllare l'**arrotondamento dei tempi di transito** e la **dimensione del grafo tempo-espanso**.

- I tempi di transito τ_e vengono **arrotondati per eccesso** al multiplo più vicino di Δ:

  $\tau'_e := \lceil \tau_e / \Delta \rceil \cdot \Delta$

- Il makespan effettivo nella rete con tempi arrotondati è:

  $T' := T + \Delta(n-1) \leq T(1+\epsilon)$

  (Nota: **la formula da te usata con (1+ε)³T/Δ è errata** – questa forma non compare nel risultato noto dell'algoritmo.)

---

2. Costruzione della rete tempo-espansa condensata

- Si costruisce la **rete Δ condensata** tempo-espansa $\mathcal{N}^{T'/\Delta}$, usando i tempi di transito arrotondati.
- La rete contiene:
  - $O(n \cdot T'/\Delta) = O(n^2/\epsilon)$ **nodi**
  - $O(m \cdot T'/\Delta) = O(nm/\epsilon)$**archi**

  Questo garantisce **complessità polinomiale** in $n, m, \frac{1}{\epsilon}$.

---

3. Calcolo del flusso statico

- Si calcola un **flusso statico** xx nella rete condensata che:
  - **soddisfa esattamente** le **domande originali D** entro makespan T'
  - ha **costo al più C**
- Se non esiste tale flusso statico, significa che **il makespan tentato T è troppo piccolo**, quindi la ricerca binaria continua.

---

4. Ricostruzione del flusso nel tempo nella rete originale

- Il flusso statico xx viene **trasformato** in un flusso dinamico $\hat{f}$ nella rete originale con tempi di transito τ\tau, attraverso **una traslazione temporale**:

  $\hat{f}_e(t) := x_e(t - i\Delta) \quad \text{per ogni arco } e = (v_i, v_j)$
- Questo flusso:
  - segue **gli stessi cammini**
  - arriva **al più** $(n-1)\Delta$ dopo rispetto al tempo originale
  - ha makespan al più $T + (n-1)\Delta \leq T(1+\epsilon)$
  - ha lo **stesso costo** e **soddisfa le stesse domande**

Questo schema permette di trovare, in **tempo polinomiale**, una **soluzione (1+ϵ)(1+\epsilon)-approssimata** al problema del quickest flow con vincolo di costo, **trasformando un problema dinamico in uno statico** grazie all'arrotondamento e al tempo-espanso condensato.

4.4 -4.6

> **Unless P=NP**, there is **no FPTAS for the quickest multicommodity flow problem when intermediate node storage is prohibited** and flow may only be sent on simple paths. If, however, intermediate node storage is allowed, then there exists an optimal solution that uses
> only simple flow paths: Inst