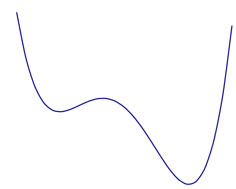
Capitolo 15

Ricerca locale

E' una tecnica di programmazione. L'idea generale è la seguente: se si conosce una soluzione ammissibile (non necessariamente ottima) ad un problema di ottimizzazione, si può cercare di trovare una soluzione migliore nelle "vicinanze" di quella precedente. Si continua in questo modo fino a quando non si è più in grado di migliorare

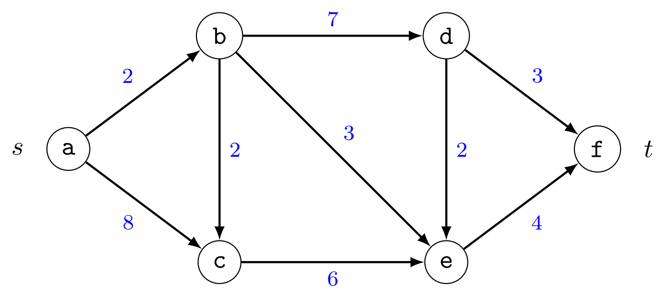
Algoritmo 1: Logica della ricerca locale

ricercaLocale
Sol = una soluzione ammissibile del problema
while $\exists S \in I(Sol)$ migliore di Sol **do**
  $Sol = S$



15.1 Problema di flusso massimo

Prima di definire il problema abbiamo bisogno di definire alcune entità e le corrispondenti proprietà.



Definizione 15.1.1 (rete di flusso)**.** Una rete di flusso $G = (V, E, s, t, c)$ è data da:

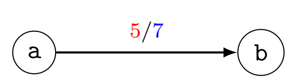
- un grafo orientato $G = (V, E)$;
- un nodo $s \in V$ detto sorgente;
- un nodo $t \in V$ detto pozzo;
- una funzione di capacità $c: V \times V \to \mathbb{R}^{\geqslant 0}$, tale per cui $(u, v) \notin R \Rightarrow c(u, v) = 0$.

Definizione 15.1.2. Un flusso in G è una funzione $f: V \times V \to \mathbb{R}$ (nota che può essere anche negativo) che gode delle seguenti proprietà:

- Vincolo sulla capacità;
- Antisimmetria;
- Conservazione del flusso.

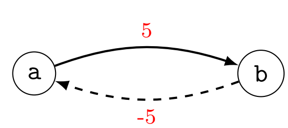
Vincolo sulla capacità Il flusso non deve eccedere la capacità sull'arco, ovvero

$$\forall u, v \in V : f(u, v) \leqslant c(u, v)$$



Antisimmetria Il flusso che passa dal nodo u al nodo v, è pari a meno il flusso dal nodo v al nodo u, ovvero

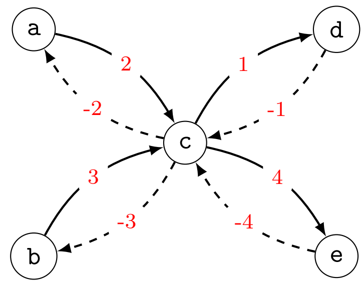
$$\forall u, v \in V : f(u, v) = -f(v, u)$$



Conservazione del flusso Per ogni nodo, la somma dei flussi entranti deve essere uguale alla somma dei flussi uscenti.

$$\forall u \in V - \{s,t\}: \sum_{v \in V} f(u,v) = 0$$

La sommatoria dei flussi che partono da u a qualunque altro nodo, ad esclusione della sorgente e del pozzo, deve essere 0.



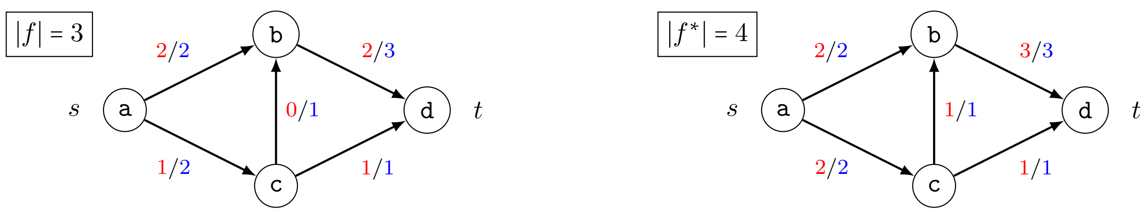
Definizione 15.1.3 (valore del flusso)**.** Il valore di un flusso f è definito come la quantità di flusso uscente da s.

$$|f| = \sum_{(s,v) \in E} f(s,v)$$

15.1.1 Problema del flusso massimo

Definizione 15.1.4 (Flusso massimo)**.** Data una rete $G = (V, E, s, t, c)$, trovare un flusso che abbia valore massimo fra tutti i flussi associabili alla rete.

$$|f^*| = \max\{\,|f|\,\}$$



(a) Il valore del flusso in questo caso è pari a 3 ma non è massimo.

(b) Il valore del flusso massimo per questa rete è 4.

Nota. Il nostro obiettivo è quindi quello di trovare il flusso massimo.

15.2 Metodo delle reti residue

Utilizziamo quindi il *metodo delle reti residue*. Il metodo consiste nel partire da un flusso "corrente" f, inizialmente nullo e 1) si "sottrae" il flusso attuale dalla rete iniziale, ottenendo così una rete residua; 2) si cerca un flusso g all'interno della rete residua; 3) si somma g ad f., ripetendo le azioni precedenti fino a quando non è più possibile trovare un flusso positivo g.

Se il procedimento viene svolto correttamente, è possibile dimostrare che questo approccio restituisce un flusso massimo.

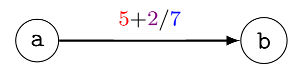
Nota. È un algoritmo di ricerca locale, senza una dimostrazione non è possibile dire che la soluzione sia ottima.

Definizione 15.2.1 (Flusso nullo)**.** Definiamo flusso nullo la funzione $f_0 \colon V \times V \to R^{\leqslant 0}$ tale che $f(u,v) = 0$ per ogni $u, v \in V$.

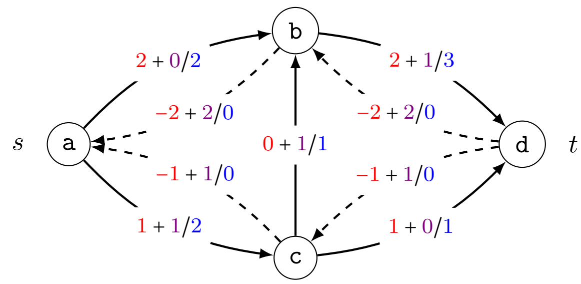
Definizione 15.2.2 (Somma di flussi)**.** Per ogni coppia di flussi f_1 e f_2 in G, definiamo il flusso somma $g = f_1 + f_2$ come un flusso tale per cui $g(u,v) = f_1(u,v) + f_2(u,v)$

Nota. È possibile invalidare il vincolo sulla capacità.

Definizione 15.2.3 (Capacità residua)**.** Definiamo capacità residua di un flusso f una funzione $c_f\colon V \times V \to R^{\geqslant 0}$ tale che $c_f(u,v) = c(u,v) - f(u,v)$
Per la definizione di capacità residua, si creano degli archi all'indietro: $c_f(b,a) = c(b,a) - f(b,a) = 0 - (-5) = 5$



Definizione 15.2.4 (Reti residue)**.** Data una rete di flusso $G = (V, E, s, t, c)$ e un flusso f su G, possiamo costruire una rete residua $G_f = (V, E_f, s, t, c_f)$, tale per cui $(u,v) \in E_f$ se e solo se $c_f(u,v) > 0$.



15.2.1 Algoritmo

Algoritmo 1: Schema generale

int[][] maxFlow(GRAPH G, NODE s, NODE t, **int**[][] c)

  $f \leftarrow f_0$ `// inizializza un flusso nullo` $r \leftarrow c$ `// capacità iniziale`

  `// Fintanto che non rimane un flusso nullo`

  repeat

    $g \leftarrow$ trova un flusso r tale che $|g| > 0$, altrimenti f_0

    $f \mathrel{+}= g$ `// necessita di una dimostrazione`

    $r \leftarrow$ rete di flusso residua del flusso f in G

  until $g == f_0$

  return f `// il flusso`

Dimostrazione di correttezza

Lemma 1. *Se f è un flusso in G e g è un flusso in G_f, allora $f + g$ è un flusso in G.*

Vincolo sulla capacità

Dimostrazione.

$$
\begin{aligned}
g(u,v) &\leqslant c_f(u,v) && \text{sostituisco } c_f\\
g(u,v) &\leqslant c(u,v) - f(u,v) && \text{aggiungo termine uguale}\\
f(u,v) + g(u,v) &\leqslant c(u,v) - \cancel{f(u,v)} + \cancel{f(u,v)} && \text{raccolgo e semplifico}\\
(f+g)(u,v) &\leqslant c_f(u,v)
\end{aligned}
$$

□

Vincolo sulla capacità `MATERIALE MANCANTE`

Esempio Fare riferimento alla spiegazione grafica (minuto 35:27) qui non riportata (riferimento trasparenze $\boxed{00}$ - $\boxed{00}$.

Algoritmo 2: Algoritmo di Ford-Fulkerson

int nomeFunzione(*parameters*)

  return 0

Ricerca del cammino Manca del testo. Notare che le due cose non si escludono a vicenda. Il costo della visita è $\mathcal{O}(V + E)$.

Codice 15.1: inserire didascalia

```java
/**
 * Compute the max-flow using the Ford-Fulkerson algorithm
 * @param C the capacity matrix
 * @param s the source node
 * @param t the sink node
 * @return the flow matrix
 */
private static int[][] flow(int[][] C, int s, int t) {
    // Create an empty flow
    int[][] F = new int[C.length][C.length];

    // Visited array to perform DFS, initially empty
    boolean[] visited = new boolean[C.length];

    // Repeat until there is no path
    while (dfs(C, F, s, t, visited, Integer.MAX_VALUE) > 0) {
        Arrays.fill(visited, false);
    }
    return F;
}
```

Codice 15.2: inserire didascalia

```java
/**
 * Performs a DFS starting from node i and trying to reach node t.
 * @param C the capacity matrix; if capacity[x][y]>0, there is a edge from x to y
 * @param F the flow matrix to be computed
 * @param i the current node,
 * @param t the sink node
 * @param visited the boolean set containing the nodes that have been visited
 * @param min the smallest capacity found during the visit.
 * @returns the value of the additional flow found during the DFS
 */
private static int dfs(int[][] C, int[][] F, int i, int t, boolean[] visited, int min) {
    // If sink has been reached, terminate
    if (i == t) return min;

    visited[i] = true;
    for (int j = 0; j < C.length; j++) {
        if (C[i][j] > 0 && !visited[j]) {
            // Non-visited neighbor
            int val = dfs(C, F, j, t, visited, Math.min(min, C[i][j]));
            if (val > 0) {
                C[i][j] = C[i][j] - val;
                C[j][i] = C[j][i] + val;

                F[i][j] = F[i][j] + val;
                F[j][i] = F[j][i] - val;

                return val;
            }
        }
    }
```

```
    // The sink has not been found
    return 0;
}
```

Complessità Se le capacità della rete sono intere, l'algoritmo di Ford e Fulkerson ha complessità $\mathcal{O}((V+E)\,|f^*|)$ (il costo della visita in ampiezza) o complessità $\mathcal{O}(V^2\,|f^*|)$ nel caso pessimo (se fatto su una matrice).

L'algoritmo di Edmonds e Karp ha complessità $\Omega(V \cdot E^2)$ nel caso pessimo.

Entrambi i limiti superiori sono validi, si prende quindi il più basso fra i due.

Nota. Il cambio di notazione (ovvero segnalare i nodi con V (*vertex*) anziché con n, e i lati con E (*edges*) anziché con m) è necessario per non fare confusione con gli esercizi.

Lemma 2. *Il numero totale di aumenti di flusso eseguiti dall'algoritmo di Edmonds e Karp è* $\mathcal{O}(V \cdot E)$.

15.2.2 Dimostrazione complessità

MATERIALE MANCANTE

15.2.3 Dimostrazione correttezza

Per fare la dimostrazione di correttezza dobbiamo (re)introdurre delle definizioni.

Definizione 15.2.5 (Taglio)**.** Un taglio (S,T) della rete di flusso $G = (V, E, s, t, c)$ è una partizione V in S e $T = V - S$ tale che $s \in S$, $t \in T$.

Mettere schema a destra.

Definizione 15.2.6 (Capacità di un taglio)**.** La capacità $c(S,T)$ attraverso il taglio S,T è pari a

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$

Mettere schema a destra.

Definizione 15.2.7 (Flusso di un taglio)**.** Se f è un flusso in G, il flusso netto $F(S,T)$ attraverso (S,T) è pari a

$$F(S,T) = \sum_{u \in S, v \in T} f(u,v)$$

Mettere schema a destra.

Lemma 3 (Valore del flusso di un taglio)**.** *Dato un flusso f e un taglio (S,T), la quantità di flusso $F(S,T)$ che attraversa il taglio è uguale a $|f|$.*

$$\begin{aligned}
F(S,T) &= \sum_{u \in S, v \in T} f(u,v) && \text{V = T − S} \\
&= \sum_{u \in S, v \in V} f(u,v) - \sum_{u,v \in S} f(u,v) && \text{conservazione del flusso} \\
&= \underbrace{\sum_{u \in S - \{s\}, v \in V} f(u,v)}_{0} + \sum_{v \in V} f(s,v) + \underbrace{\sum_{u,v \in S} f(u,v)}_{0} && \text{e antisimmetria} \\
& && \text{per definizione} \\
&= \sum_{v \in V} f(s,v) = f
\end{aligned}$$

Un taglio definisce un limite superiorie al flusso massimo che può essere presente nella rete.

Lemma 4 (Capacità del taglio)**.** *Il flusso massimo è limitato superiormente dalla capacità del taglio minimo (ovvero il taglio la cui capacità è minotr fra tutti i taglio).*

MATERIALE MANCANTE

La somma dei flussi dev'essere necessariamente minore della somma delle capacità.

MATERIALE MANCANTE

Quindi, il valore del flisso è limitato superiormente dalla capacità di tutti i possibili tagli.

Teorema 5 (Teorema del flusso massimo)**.** *Le seguenti affermazioni sono equivalenti:*

- *a*

- b
- c

`DA FINIRE`