Project Scheduling con risorse limitate

1. Classificazione delle risorse

- Classificazione delle risorse in base alla disponibilità.

  - *Risorse rinnovabili*

    Sono risorse utilizzate per l'esecuzione di una attività per tutta la sua durata, ma sono rese nuovamente disponibili non appena l'attività è terminata. Sono pertanto risorse per le quali si hanno vincoli sull'uso complessivo ad ogni istante della durata del progetto.

    Esempi: personale, macchine, flusso di denaro, potenza, flusso di carburante.

  - *Risorse non rinnovabili*

    Sono risorse per le quali il vincolo sulla disponibilità coinvolge solo il consumo totale per l'intera durata del progetto. Si tratta quindi di risorse disponibili e consumate sulla base dell'intero progetto.

    Esempi: denaro limitato nel budget, energia, carburante.

  - *Risorse doppiamente vincolate*

    Sono risorse simultaneamente vincolate sia in termini di uso complessivo ad ogni istante che in termini di consumo totale per l'intero progetto.

Project Scheduling con risorse limitate

2. Considerazioni preliminari

- il *CPM* e il *PERT* sono tecniche basate sull'ipotesi che le risorse addizionali non esistono (o sono illimitate) pertanto *non sono tecniche idonee* a risolvere problemi di project scheduling con vincoli sulle risorse.

- In generale, quando si ha a che fare con *vincoli sulle risorse* i problemi di project scheduling sono *NP-hard in senso forte*. Sono una *generalizzazione* dei problemi di *job-shop*.

- Pertanto il primo passo da fare nell'analisi di tali problemi è quello di sviluppare un *efficiente modello teorico* che ne rappresenti le *caratteristiche di base* attraverso opportune *strutture matematiche* relativamente semplici che permettano di poter effettuare delle *analisi teoriche* (come: *rappresentazione della spazio delle soluzioni, rilassamenti, criteri di dominanza, ecc.*) e che siano adatte al *disegno di algoritmi di risoluzione sia euristici che esatti*.

- Ciò può essere ottenuto attraverso modelli basati su *grafi non orientati* nel caso di attività indipendenti (cioè senza vincoli di precedenza), o su *grafi orientati* rappresentanti *insiemi parzialmente ordinati* nel caso di attività con vincoli di precendenza.

Project Scheduling con risorse limitate

3. Definizioni parametri

- Faremo riferimento da ora in poi a problemi di project scheduling con *solo risorse rinnovabili (limitate)*. Siano:

  - *Parametri*

    - $\mathcal{T} = \{T_1, \ldots, T_n\}$ l'insieme di n *attività* (*task*) che devono essere eseguite *senza interruzione*
    - $p = (p_1, \ldots, p_n)$ il vettore delle *durate* dei task
    - $R_0 \subseteq \mathcal{T} \times \mathcal{T}$ un *ordinamento parziale* definito su $\mathcal{T}$ che descrive le relazioni (vincoli) di precedenza (diretta e indiretta) sui task di tipo finish-to-start;
    - $\mathcal{R} = \{R_1, \ldots, R_s\}$ un insieme di s *risorse rinnovabili* con:
      r_{ih} rappresentante le unità di risorsa R_h utilizzate dal task T_i durante ogni istante in cui è in esecuzione;
      Q_h rappresentante la *quantità massima* (in unità di risorsa) di *risorsa* R_h *disponibile* in ogni istante
    - D *limite* sulla *durata* del *progetto*

  - *Variabili*

    - $(s_1, \ldots, s_n)$ vettore degli *istanti di inizio* delle attività. Tale vettore è completamente specificato dalla *schedula* $S = (s_1(S), \ldots, s_n(S))$ delle attività del progetto.

Project Scheduling con risorse limitate

4. Considerazioni introduttive

- Due task T_i, T_j che possono essere eseguiti *contemporaneamente* (perchè non vincolati da relazione di precedenza) *richiedono* complessivamente $r_{ih} + r_{jh}$ unità di risorsa R_h se vengono *eseguiti simultaneamente*.

- Pertanto il vincolo sulle risorse limitate potrebbe imporre di *sequenziare* (cioè aggiungere relazioni di precedenza) attività che utilizzano la stessa risorsa, provocando quindi *un incremento del tempo di completamento* di qualche attività che può comportare un *aumento della durata del progetto*.

- Viceversa, *eseguendo contemporaneamente* delle attività per ridurre la durata del progetto si può produrre un *incremento dell'uso delle risorse*.

- Siccome i task devono essere eseguiti senza interruzione, ogni possibile *misura di prestazione* di una determinata scelta sull'esecuzione delle attività di un progetto (soluzione) *dipende dalla schedula* $S = (s_1(S), \ldots, s_n(S))$ delle attività, cioè solo dagli istanti di partenza s_i.

Project Scheduling con risorse limitate

4. Considerazioni introduttive *(continua)*

- La *prestazione* di una *schedula* S è di solito espressa attraverso *due* differenti tipi di *misure* che esprimono rispettivamente gli *aspetti temporali* e gli *aspetti legati alle risorse*.

  Aspetti temporali

  - Si fa riferimento al *tempo di completamento* $C_i(S)$ della generica attività T_i per la schedula S.

    $$C_i(S) = s_i(S) + p_i$$

  - Volendo ancora rappresentare la prestazione di una schedula S secondo un unico *indice di prestazione temporale* si ricorre di solito ad una *funzione regolare* $\tau(C_1(S), \ldots, C_n(S))$.

    Cioè una funzione non decrescente $\tau : \mathbb{R}^n \rightarrow \mathbb{R}^1$ dei tempi di completamento C_i

    $\tau(C_1(S), \ldots, C_n(S))$ rappresenta il *costo di esecuzione* della schedula S.

    Ad esempio.

    $$\tau(C_1(S), \ldots, C_n(S)) = \max [C_i(S)] = C_{\max}(S)$$

    è la *durata del progetto* (o *makespan della schedula*).

Project Scheduling con risorse limitate

4. Considerazioni introduttive (continua)

Aspetti legati all'uso delle risorse

- Data una schedula S si fa riferimento all'*ammontare massimo* $r_h^{\max}(S)$ di risorsa R_h (per $h = 1, \ldots, s$) utilizzata durante i vari istanti in cui il progetto è in corso

$$r_h^{\max}(S) = \max \{\Sigma_{\{i:\, s_i(S) < \theta \le s_i(S) + p_i\}} r_{ih} \mid \theta = 1, 2, \ldots, T\}.$$

- Analogamente, volendo rappresentare la prestazione di una schedula S, secondo un unico *indice di prestazione sull'uso delle risorse utilizzate*, si fa ricorso ad una *funzione non decrescente* $\kappa : \mathbb{R}^s \to \mathbb{R}^1$

$$\kappa(r_1^{\max}(S), \ldots, r_s^{\max}(S)),$$

rappresentante il *costo delle risorse* di S.

- Ad esempio:

$$\kappa(r_1^{\max}(S), \ldots, r_s^{\max}(S)) = \Sigma^s_{h=1} K_h\, r_h^{\max}(S)$$

dove K_h è il costo per unità di risorsa R_h.

Project Scheduling con risorse limitate

4. Due principali problemi

Resource Constrained Proj. Sched. Problem (RCPSP)
(o *problema delle risorse limitate*)

- Dato un ordinamento parziale R_0 sulle attività $\mathcal{T}$ e un limite Q_h sulla disponibiltà della risorsa R_h, determinare la schedula di progetto S^* che minimizza il costo di esecuzione $\tau(C_1(S), \ldots, C_n(S))$ tra tutte le schedule ammissibili $S \in \mathcal{S}$, cioè per le quali $r_h^{\max}(S) \leq Q_h$, $h=1, \ldots, s$, e che rispetta l'ordinamento parziale R_0.

Resource Availability Cost Problem (RACP)
(o *problema del tempo limitato*)

- Dato un ordinamento parziale R_0 sulle attività $\mathcal{T}$ e un limite Q_h sulla disponibiltà della risorsa R_h, determinare la schedula di progetto S^* che minimizza il costo di delle risorse $\kappa(r_1^{\max}(S), \ldots, r_s^{\max}(S))$ tra tutte le schedule ammissibili $S \in \mathcal{S}$, cioè per le quali $C_{\max}(S) \leq D$ e che rispetta l'ordinamento parziale R_0.

Project Scheduling con risorse limitate

5.1. RCPSP(C_{max}): Formulazione

- Consideriamo *RCPSP* con l'obiettivo di minimizzare il tempo di completamento del progetto *RCPSP(C_{max})*.

- Senza perdita di generalità supponiamo che T_1 sia l'unico task (eventualmente fittizio) a non avere predecessori e T_n sia l'unico task (eventualmente fittizio) a non avere successori, e sia $G_0 = (\mathcal{T}, R_0)$ il *grafo di progetto con attività sui nodi*.

Formulazione concettuale

$$C_n^* = p_n + \min s_n$$

s.t.

$$s_j - s_i \geq p_j, \qquad \forall\ (T_i, T_j) \in R_0$$

$$s_1 = 0$$

$$\sum_{T_i \in S_t} r_{ih} \leq Q_h,\ h = 1, \ldots, s;\ t = 1, \ldots, T$$

dove $S_t = \{T_i \in \mathcal{T}: s_i < t \leq s_i + p_i\}$ è l'insieme delle attività in esecuzione nel periodo unitario t (cioè nell'intervallo $[t-1, t]$) e T è un UB di C_n^*.

- Non è una formulazione agevole perché S_t dipende dalla soluzione stessa.

Project Scheduling con risorse limitate

5.1. RCPSP(C_{max}): Formulazione (continua)

Formulazione di Pritsker et al. (1969)

- Consideriamo la variabile binaria x_{it}, con $x_{it} = 1$ se T_i inizia al tempo t e $x_{it} = 0$ altrimenti.
- Ovviamente t può essere ristretto ai tempi ammissibili di inizio per T_i, cioè $t \in \{ES_i, LS_i\}$, determinabili con le ricorsioni in avanti e indietro, assumendo $LF_n = T$ con $T = UB$ sulla durata minima del progetto (es., $UB = \sum_{T_i \in \tau} p_i$).

$$p_n + \min \sum_{t=ES_n}^{LS_n} t \cdot x_{nt} \qquad (1)$$

s.t.

$$\sum_{t=ES_i}^{LS_i} x_{it} = 1, \qquad i = 1,\ldots,n \qquad (2)$$

$$\sum_{t=ES_i}^{LS_i} t \cdot x_{it} \le \sum_{t=ES_j}^{LS_j} t \cdot x_{jt} - p_j, \quad \forall (T_i, T_j) \in R_0 \qquad (3)$$

$$\sum_{i=1}^{n} r_{ih} \sum_{q=\max\{t-p_i, ES_i\}}^{\min\{t-1, LS_i\}} x_{iq} = Q_h, \qquad h = 1,\ldots,s; t = 1,\ldots,T \qquad (4)$$

$$x_{it} \in \{0,1\}, \qquad i = 1,\ldots,n; t = ES_i,\ldots,LS_i \qquad (5)$$

- N.B.: Le attività T_i in esecuzione nel periodo unitario t sono quelle i cui s_i cadono nell'intervallo $[t-p_i, t-1]$ (oltre ovviamente a appartenere all'intervallo $[ES_i, LS_i]$).
- Quindi T_i è in esecuzione nel periodo unitario t se $\max\{t-p_i, ES_i\} \le s_i \le \min\{t-1, LS_i\}$.

Project Scheduling con risorse limitate

5.2. RCPSP(C_{max}): Lower bound

LB basato sul cammino critico

- Si ignorano i vincoli (4) sulla limitatezza delle risorse, riconducendo il problema al caso con risorse illimitate.

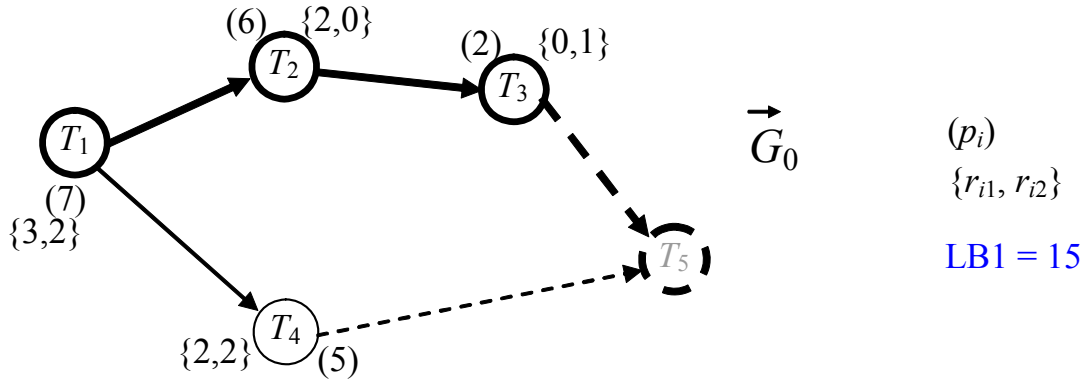
Lower bound: LB1

- *1° ipotesi: risorse tutte in quantità illimitata.*

  LB1 $= l_\pi$ lunghezza del cammino critico π del grafo $\vec{G}_0 = (T, R_0)$ delle precedenze.

Esempio

- 4 task $T = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.
- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.



Project Scheduling con risorse limitate

5.2. RCPSP(C_{max}): Lower bound

LB basato sulla sequenza critica (Stinson et al., 1978)

- Miglioramento del LB precedente.
- Si determina un cammino critico π ipotizzando di rilassare il vincolo (4) sulla limitatezza delle risorse. Sia l_π la sua lunghezza (n.b.: $\text{LB1} = l_\pi$).
- Sia $\mathcal{T}_\pi$ l'insieme della attività del cammino critico.
- Per ciascuna attività $T_i \in \mathcal{T} \setminus \mathcal{T}_\pi$ si valuta la (massima) durata di tempo $l_i \leq p_i$ in cui T_i nell'intervallo $[ES_i, LF_i]$ può essere eseguita (senza interruzione) in parallelo alle attività del cammino critico π.
- Per eseguire le attività $\mathcal{T}_\pi \cup \{T_i\}$ occorre un tempo pari a $l_\pi + (p_i - l_i)$ che pertanto costituisce la lunghezza della *sequenza critica* formata da tali attività.
- Il massimo tra le lunghezze delle sequenze critiche ottenute per ogni attività $T_i \in \mathcal{T} \setminus \mathcal{T}_\pi$ è un valido LB

Lower bound: LBs

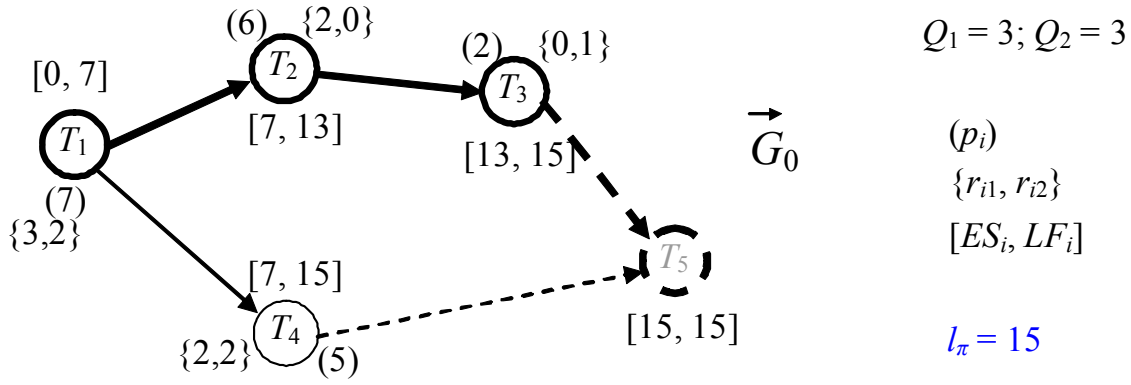
$$\text{LBs} = l_\pi + \max_{T_i \in \mathcal{T} \setminus \mathcal{T}_\pi} \{p_i - l_i\}.$$

Project Scheduling con risorse limitate

5.2. RCPSP(C_{max}): Lower bound

Esempio

- 4 task $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.
- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.



- Per l'unica attività $T_4 \in \mathcal{T} \setminus \mathcal{T}_\pi$ si ha che $l_4 = 2 \leq p_4 = 5$.
- Quindi $LBs = l_\pi + \max_{T_i \in \mathcal{T} \setminus \mathcal{T}_\pi} \{p_i - l_i\} = 15 + (5 - 2) = 18$.

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

LB basato sulle risorse

- L'attività T_i richiede un ammontare totale di risorsa R_h pari $r_{ih}\, p_i$.

- Pertanto $\sum_{T_i \in \tau} r_{ih}\, p_i$ è la quantità totale di risorsa R_h richiesta durante il progetto.

- In ogni periodo unitario di tempo la risorsa R_h è disponibile in quantità pari a Q_h.

- È quindi necessario almeno un tempo pari a $\lceil (\sum_{T_i \in \tau} r_{ih}\, p_i)/Q_h \rceil$ per eseguire tutte le attività.

- Dovendo valere ciò per ogni risorsa R_h (per $h = 1, \ldots, s$) si ha:

Lower bound: LBr

$$\text{LBr} = \max_h \{\lceil (\sum_{T_i \in \tau} r_{ih}\, p_i)/Q_h \rceil\}$$

Project Scheduling con risorse limitate

5.2. RCPSP(C_{max}): Lower bound

Esempio

- 4 task $T = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.
- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.

i	p_i	$r_{i1}; r_{i2}$	$r_{i1}\, p_i$	$r_{i2}\, p_i$
1	7	3; 2	21	14
2	6	2; 0	12	0
3	2	0; 2	0	4
4	5	2; 2	10	10
$\sum_i$	20	-	43	28
$\lceil (\sum_i r_{ih}\, p_i)/Q_h \rceil$	-	-	15	10

- Quindi $LBr = \max_h \{\lceil (\sum_{T_i \in T} r_{ih}\, p_i)/Q_h \rceil\} = \max\{15; 10\} = 15$.

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

- Sono *euristiche costruttive* che si basano su:
  i. *schema di scheduling*
  ii. *regola di priorità*

- Lo *schema di scheduling* definisce il modo con cui la schedula ammissibile è costruita, estendendo una schedula parziale.

- Tipicamente gli schemi di scheduling sono di due tipi:
  a) *schema di scheduling seriale*
  b) *schema di scheduling parallelo*

- La *regola di priorità* , definita dalla coppia (v, extremum), definisce l'ordine di priorità nella schedulazione delle attività; permette di scegliere una o più attività da un insieme detto di decisione.

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

Schema di scheduling seriale

- Partendo dalla schedula vuota, si aggiungono sequenzialmente le attività alla schedula parziale.

- È un algoritmo iterativo che ad ogni iterazione seleziona un'attività (in base alla regola di priorità impiegata) non schedulata e la aggiunge alla corrente schedula parziale.

- All'iterazione k si considera una partizione delle attività $\mathcal{T}$ in 3 sottoinsiemi:
    - C_k (*complete set*) contenente le attività già schedulate e che quindi appartengono alla schedula parziale
    - $\mathcal{D}_k$ (*decision set*) contenente le attività non schedulate i cui predecessori diretti sono in C_k
    - $\mathcal{R}_k$ (*remaining set*) contenente le rimanenti attività

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

Schema di scheduling seriale

- All'iterazione k:
  1. Si seleziona un'attività $T_i \in \mathcal{D}_k$ a priorità più elevata e la si *schedula all'istante di tempo minimo ammissibile*, rispetto alle precedenze e alle risorse usate dalla schedula parziale;
  2. Si rimuove T_i da $\mathcal{D}_k$ e si aggiunge a C_k;
  3. Tutte le attività di $\mathcal{R}_k$ le cui attività predecessori (diretti) sono state schedulate vengono rimosse da $\mathcal{R}_k$ e aggiunte a $\mathcal{D}_k$.
- L'algoritmo termina quando tutte le attività sono in C_k.

- Al passo 1 la selezione avviene tenendo conto del valore di priorità v_i assegnato al task T_i. In particolare si seleziona T_{i*} tale che:

$$v_{i*} = \text{extremum}_{T_i \in \mathcal{D}_k} \{v_i\}$$

- È *garantito* che la *schedula generata è ammissibile*.
- La *schedula* generata è *attiva*.
- In caso di assenza di vincoli sulla limitatezza delle risorse la schedula fornita è ottima.

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

- Le *regole di priorità* possono essere classificate in:
  - regole basate sul *tempo*;
  - regole basate sulle *precedenze*;
  - regole basate sulle *risorse*.

Esempi di regole basate sul *tempo*:

- *Shortest Proc. Time* (SPT) $\Rightarrow$ (p_i, min)
- *Latest Start Time* (LST) $\Rightarrow$ (LS_i, min)
- *Latest Finish Time* (LFT) $\Rightarrow$ (LF_i, min)

Esempi di regole basate sulle *precedenze*:

- *Most Immediate Succ.* (MIS) $\Rightarrow$ ($|S_i|$, max)
- *Most Total Succ.* (MTS) $\Rightarrow$ ($|\overline{S}_i|$, max)

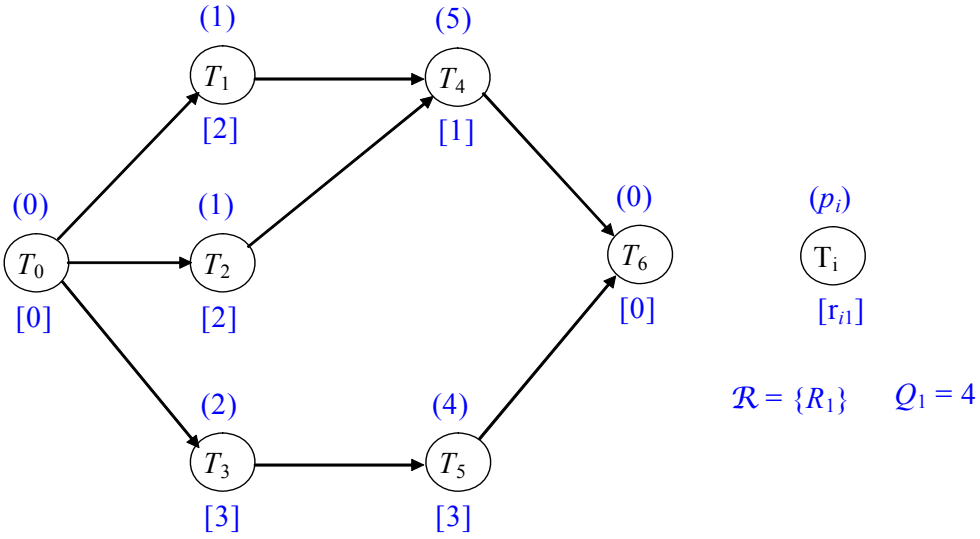
Esempi di regole basate sulle *risorse*:

- *Great Resource Demand* (GRD) $\Rightarrow$ ($p_i \sum_h r_{ih}$), max)
- *Relative Resource Usage* (RRU) $\Rightarrow$ ($\sum_h r_{ih}/Q_h$, max)

Project Scheduling con risorse limitate

5.3. RCPSP(C_{max}): Euristiche con regole di priorità

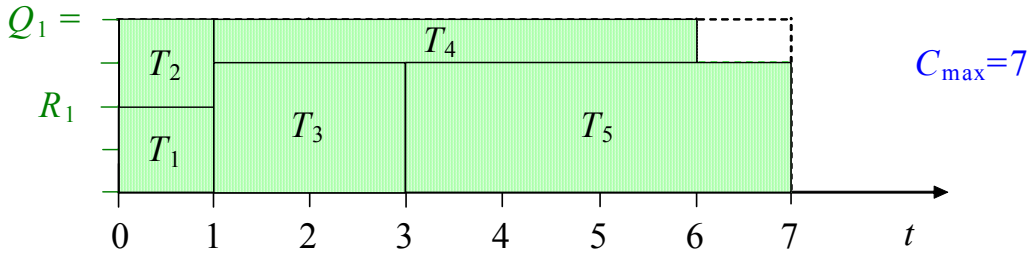
Esempio



- Si assume di aver assegnato le seguenti priorità v, con extremum = min.

T_i	T_0	T_1	T_2	T_3	T_4	T_5	T_6
v_i	0	1	1	2	6	6	6

- L'applicazione dell'algoritmo seriale produce la seguente schedula.



- Si noti che LBr = $\lceil (0\cdot0+1\cdot2+1\cdot2+2\cdot3+5\cdot1+4\cdot3+0\cdot0)/4 \rceil = 7$ (LB1 = 6; LBs = 7 scegliendo (T_0, T_3, T_5, T_6) come percorso critico), pertanto la schedula è ottima.

Project Scheduling con risorse limitate

5.4. RCPSP(C_{max}): Approccio su grafo

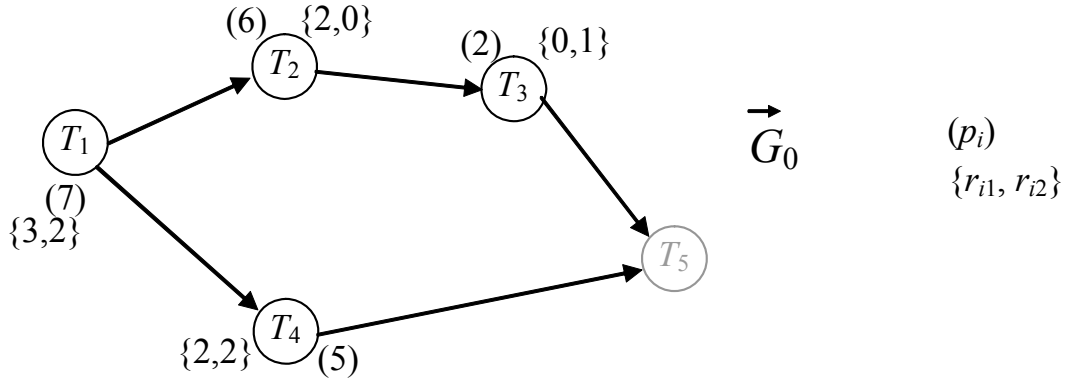
- Consideriamo *RCPSP* con l'obiettivo di minimizzare il tempo di completamento del progetto *RCPSP(C_{max})*.

- Consideriamo il *grafo di progetto con attività sui nodi* $\vec{G}_0 = (T, R_0)$ (aggiungendo eventualmente due *dummy task*: T_0 che precede tutti i task e T_{n+1} che sussegue tutti i task). $\vec{G}_0$ è il *grafo delle precedenze*.

Esempio

- 4 task $T = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.

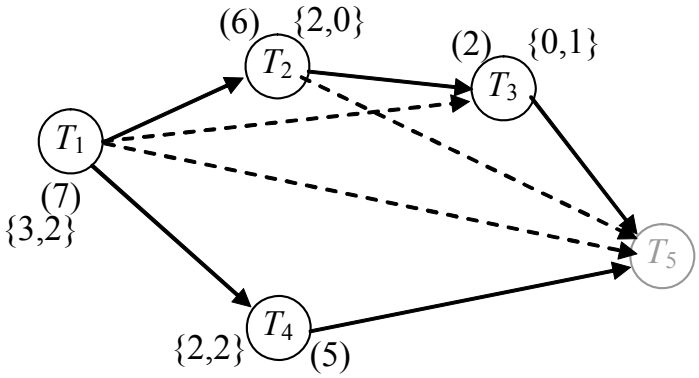
- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.



Grafo di progetto $\vec{G}_0$

Project Scheduling con risorse limitate

5.4. RCPSP(C_{max}): Approccio su grafo (continua)



$\vec{G}_{0t}$ (p_i) $\{r_{i1}, r_{i2}\}$

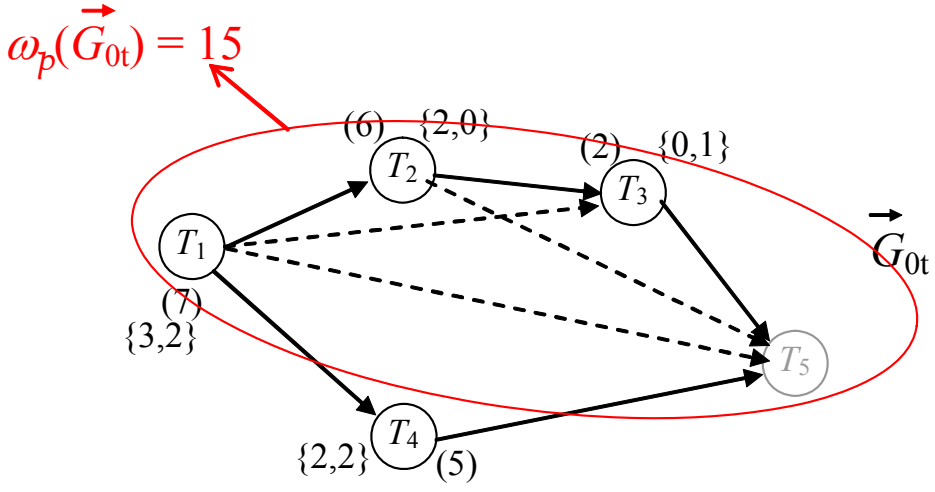
Chiusura transitiva $\vec{G}_{0t}$ del grafo di progetto

Teorema 1

- La durata minima del progetto, supponendo le risorse illimitate, è pari al *peso* $\omega_p(\vec{G}_{0t})$ *della massima clique del grafo di progetto chiuso transitivamente* $\vec{G}_{0t}$ con vertici di peso p_i.

Corollario

- La lunghezza l_π del cammino critico π sul grafo delle precedenze $\vec{G}_0 = (T, R_0)$ è pari a $\omega_p(\vec{G}_{0t})$
- $LB1 = l_\pi = \omega_p(\vec{G}_{0t})$

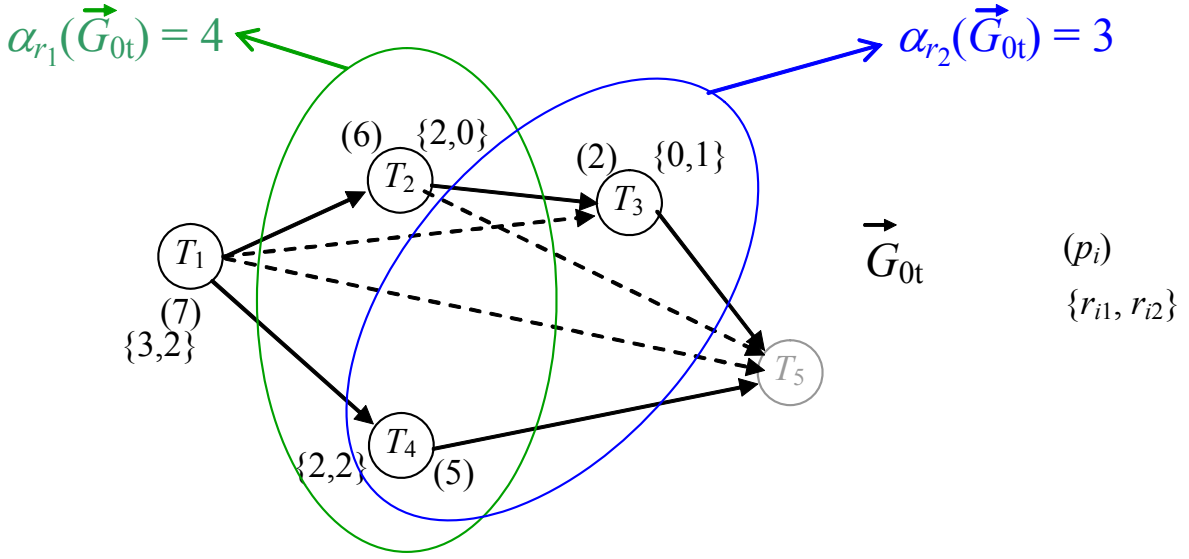


$\vec{G}_{0t}$ (p_i) $\{r_{i1}, r_{i2}\}$

5.4. RCPSP(C_{max}): Approccio su grafo (continua)

Teorema 2

- L'ammontare massimo r_h^{max} di risorsa R_h utilizzata durante i vari istanti di vita del progetto è pari al *peso* $\alpha_{r_h}(\vec{G}_{0t})$ *del massimo insieme indipendente del grafo di progetto chiuso transitivamente* $\vec{G}_{0t}$ con vertici di peso r_{ih}.



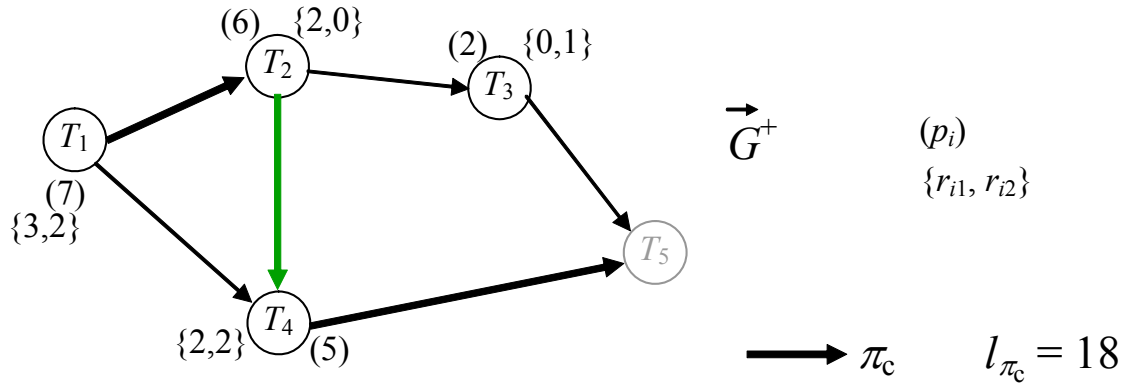
- Una *soluzione ammissibile* è rappresentata dal grafo diretto aciclico $\vec{G}^+ = (T, R)$ ottenuto da $\vec{G}_0$ aggiungendo archi (cioè precedenze, $R_0 \subseteq R$) in modo che per la sua chiusura transitiva $\vec{G}_t^+$ si ha:

$$\alpha_{r_h}(\vec{G}_t^+) = r_h^{max} \leq Q_h, \qquad h = 1, \ldots, s$$

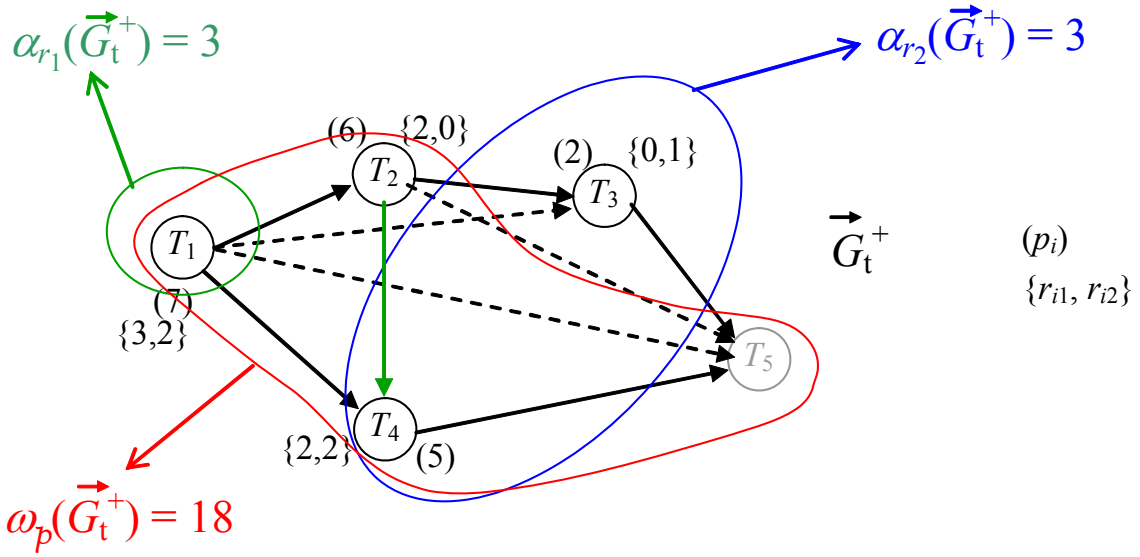
Project Scheduling con risorse limitate

5.4. RCPSP(C_{max}): Approccio su grafo (continua)

- Chiamiamo $\vec{G}^+$ *grafo esteso ammissibile* di $\vec{G}_0$.



la sua chiusura transitiva è $\vec{G}_t^+$ e $\alpha_{r_h}(\vec{G}_t^+) = \leq Q_h, \ \forall\ h$.



- La durata minima del progetto associato alla soluzione $\vec{G}^+$ *grafo esteso ammissibile* di $\vec{G}_0$ è pari:
  - o al *peso* $\omega_p(G_t^+)$ *della massima clique del grafo* $\vec{G}_t^+$ *chiusura transitiva di* $\vec{G}^+$ con vertici di peso p_i.

  ovvero
  - o alla lunghezza $l_{\pi_c}(\vec{G}^+)$ del cammino critico π_c (cioè di lunghezza massima) di $\vec{G}^+$.

Project Scheduling con risorse limitate

5.4. RCPSP(C_{max}): Approccio su grafo (continua)

- Dato il grafo diretto aciclico $\vec{G}^+ = (T, R)$ insieme al vettore delle durate p dei task, si definisce

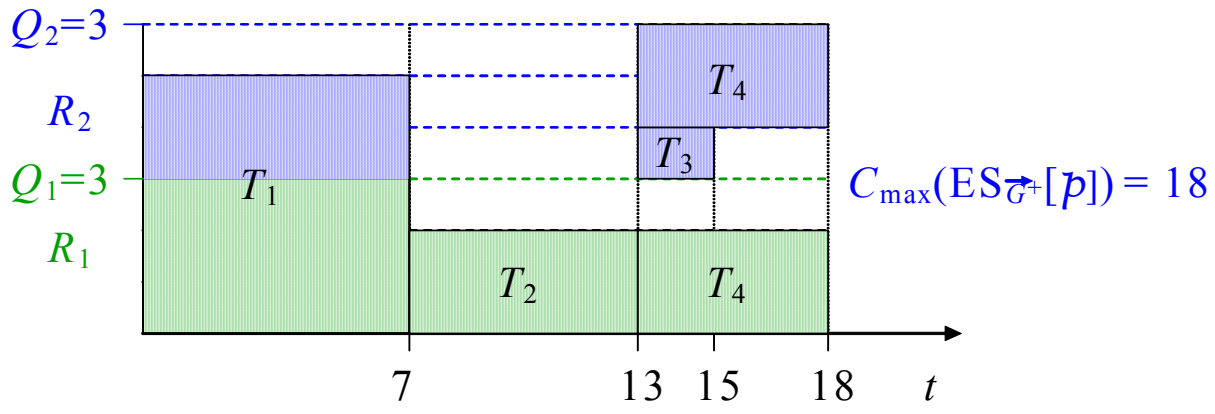
Definizione

- *Earliest start schedule* $\mathrm{ES}_{\vec{G}^+}[p]$ indotta da $\vec{G}^+$ e p la schedula con i seguenti starting time per i task:

$$t_j(\mathrm{ES}_{\vec{G}^+}[p]) = \begin{cases} 0 \text{ se } T_j \textit{ non ha predecessori} \text{ in } \vec{G}^+ \\ \max_{\{T_i:\,(T_i,\,T_j)\in R\}} [t_i(\mathrm{ES}_{\vec{G}^+}[p]) + p_i] \text{ altrimenti} \end{cases}$$

- $\mathrm{ES}_{\vec{G}^+}[p]$ è una *schedula semi-attiva* poiché nessun task può essere completato prima senza violare la relazione R. Si tratta della miglior schedula associata a $\vec{G}^+$.

- Il suo *makespan* $C_{max}(\mathrm{ES}_{\vec{G}^+}[p])$ è pari alla lunghezza $l_{\pi_c}(\vec{G}^+)$ del cammino critico π_c (cioè di lunghezza massima) di $\vec{G}^+$.



Project Scheduling con risorse limitate

5.4. RCPSP(C_{max}): Approccio su grafo (continua)

- Pertanto risolvere il problema *RCPSP(C_{max})* corrisponde a determinare un grafo (esteso) $\vec{G}^*$ che estende il grafo delle precedenze $\vec{G}_0$, aggiungendo a questo archi orientati, tale che:
  - l'*estensione sia ammissibile*: il grafo esteso $\vec{G}^*$ sia aciclico e per la sua chiusura transitiva $\vec{G}_t^*$ si ha:

$$\alpha_{r_h}(\vec{G}_t^*) = r_h^{\max} \leq Q_h, \qquad h = 1, \ldots, s$$

  - la *lunghezza $l_{\pi_c}(\vec{G}^*)$ del cammino critico π_c (di lunghezza massima)* del grafo esteso $\vec{G}^*$ sia *minima*.

- Un tale grafo esteso $\vec{G}^*$ è *ottimo* e la *earliest start schedule* $ES_{\vec{G}^*}[p]$ indotta è una *schedula ottima* per *RCPSP(C_{max})*.

- Il *makespan* C^*_{max} della *schedula ottima* $ES_{\vec{G}^*}[p]$ associata è pari alla lunghezza $l_{\pi_c}(\vec{G}^*)$ del cammino critico π_c di $\vec{G}^*$.

Project Scheduling con risorse limitate

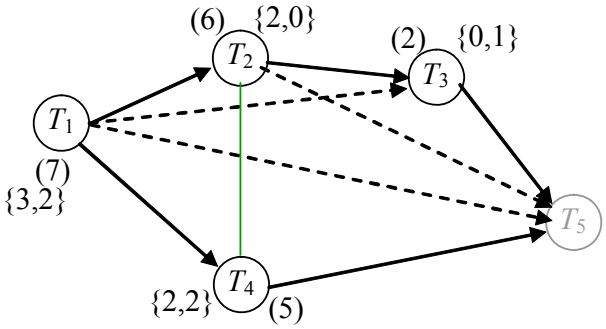
5.4. RCPSP(C_{max}): Approccio su grafo (continua)

- L'approccio su grafo descritto per l'*RCPSP(C_{max})* in pratica estende quello per il *job shop* scheduling $Jm||C_{max}$, *estendendo il concetto di grafo disgiuntivo*.

Definizione

- Il *grafo disgiuntivo dei conflitti* è il grafo $C_0 = (\mathcal{T}, A[\vec{G}_{0t}] \cup E_0)$, dove $\vec{G}_{0t}$ è la ch. trans. di $\vec{G}_0$ e

$(T_i, T_j) \in E_0$ <==> T_i e T_j *non possono essere eseguiti simultaneamente per mancanza di risorse*



C_0 (p_i) $\{r_{i1}, r_{i2}\}$

- C_0 si ottiene dalla chiusura transitiva $\vec{G}_{0t}$ di $\vec{G}_0$ e aggiungendo gli archi *non orientati* E_0, dove $(T_i, T_j) \in E_0$ se esiste una risorsa R_h per cui $r_{ih} + r_{ih} > Q_h$.
- Una *soluzione ammissibile* è rappresentata dal grafo diretto aciclico $\vec{G}^+ = (\mathcal{T}, A[\vec{G}_{0t}] \cup \vec{E}_0 \cup A^+)$ ottenuto da C_0 orientando gli archi indiretti E_0 e eventualmente aggiungendo archi in modo che per la sua chiusura transitiva $\vec{G}_t^+$ si ha:

$$\alpha_{r_h}(\vec{G}_t^+) = r_h^{\max} \le Q_h, \qquad h = 1, \ldots, s$$

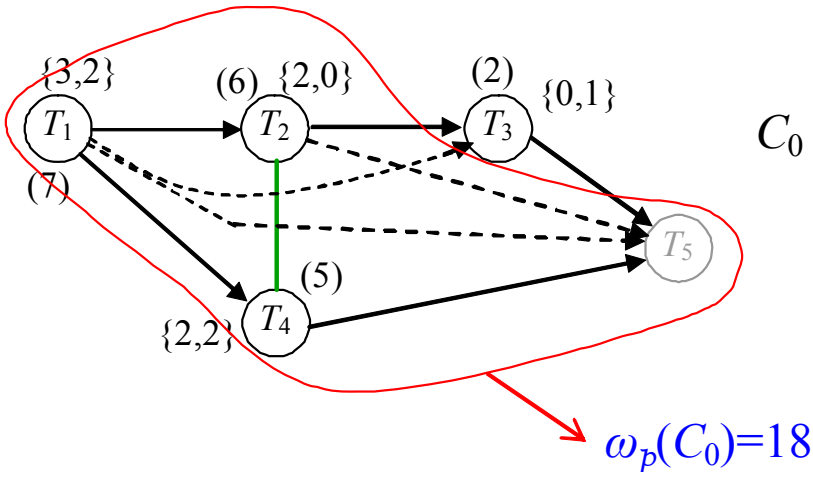
Project Scheduling con risorse limitate

5.5. RCPSP(C_{max}): Altri lower bound

Lower bound: LB2

- *Un valido LB è pari al tempo richiesto per sequenziare task mutuamente in conflitto.*
  - o task mutuamente in conflitto sono *clique* del grafo dei conflitti
  - o per eseguire task mutuamente in conflitto occorre un tempo pari alla somma delle loro durate

LB2 = *peso* $\omega_p(C_0)$ *della clique di peso massimo del grafo dei conflitti* C_0.



$Q_1 = 3; Q_2 = 3$

(p_i)

$\{r_{i1}, r_{i2}\}$

LB2 = 18

Teorema

- LB2 ≥ LB1

Project Scheduling con risorse limitate

5.5. RCPSP(C_{max}): Altri lower bound (continua)

Lower bound: LB3

- *3° ipotesi: risorse tutte tranne una in quant. illimit.*

  $LB3 = LB1 + \max_h (\delta_h)$, dove δ_h è il minimo tempo addizionale necessario per sequenziare i task che fanno uso della risorsa R_h, tale che $r_h^{max} \leq Q_h$.

N.B.:

- E' ovvio che $LB3 \geq LB1$, mentre nulla si può dire in generale tra LBs, LB2 e LB3.

Calcolo di δ_h

- Per ogni task T_i richiedente R_h, siano definiti il suo *earliest start time* ES_i e il suo *latest finish time* LF_i nell'*ipotesi 1°*.

- Il tempo addizionale richiesto dal task T_i è pari alla sua *tardiness* rispetto a LF_i mentre è ovvio che il task non può iniziare prima di ES_i.

- Quindi il *tempo addizionale* necessario per sequenziare i task richiedenti R_h, è pari alla *massima tardiness* dei task.

Project Scheduling con risorse limitate

5.5. RCPSP(C_{max}): Altri lower bound (continua)

- Pertanto il *minimo tempo addizionale* necessario δ_h è pari al *minimo valore* possibile per la *massima tardiness*.

- Il task T_i può essere modellato come un job i con *release date* $r_i = ES_i$ e *due date* $d_i = LF_i$ e richiedente r_{ih} macchine contemporaneamente.

- Inoltre tra i job sono date delle *precedenze* (quelle di R_0) e il *numero delle macchine* (parallele identiche) a disposizione è $m = Q_h$.

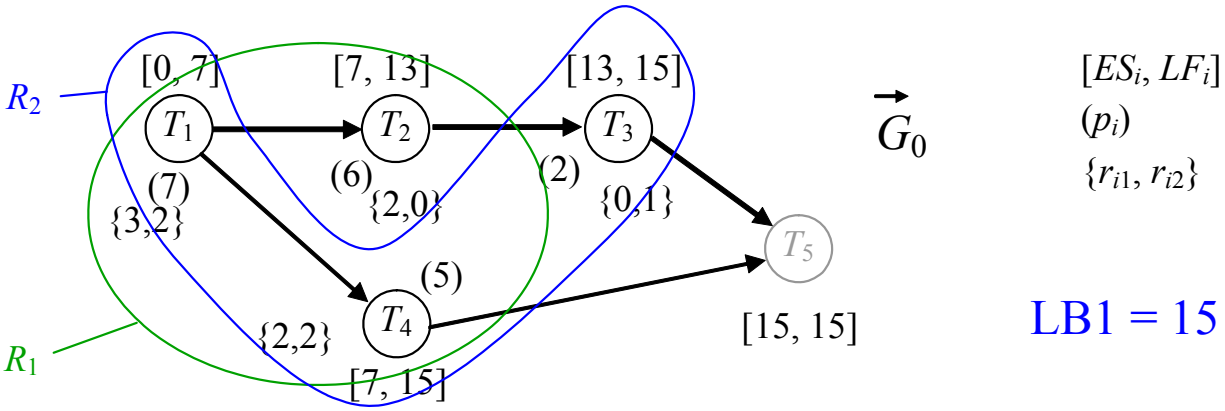
- Determinare δ_h equivale quindi a risolvere il problema $Pm|r_i, \text{ size}, \text{ prec}|L_{max}$, cioè un problema di scheduling con m *macchine parallele identiche*, con *release date*, *due date* e *precedenze*, in cui *ciascun job i richiede simultaneamente r_{ih} macchine* e l'obiettivo è *minimizzare la lateness massima*.
  Sia L^*_{max} il valore ottimo, si ha che $\delta_h = \max [0, L^*_{max}]$.

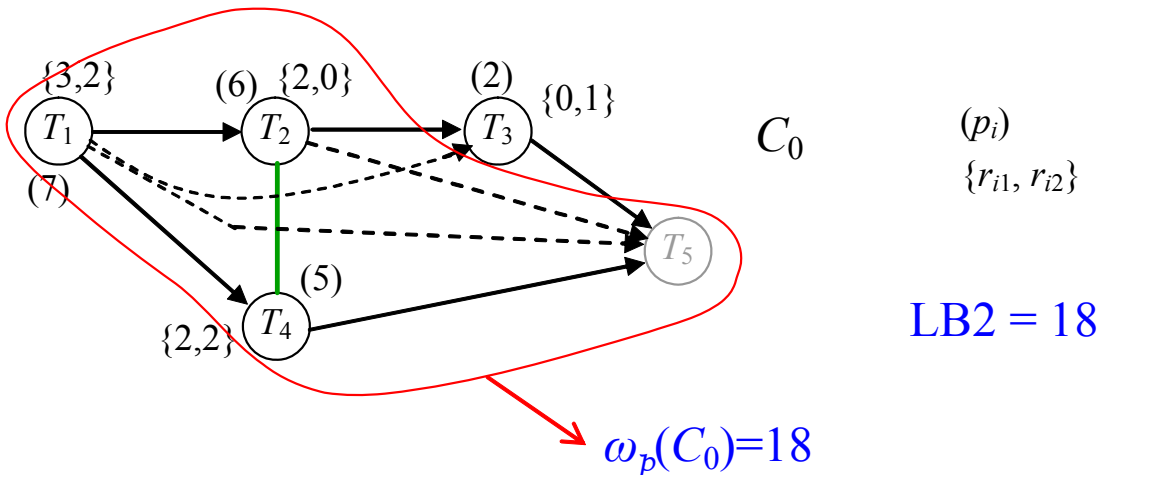
Project Scheduling con risorse limitate

5.5. RCPSP(C_{max}): Altri lower bound (continua)

Esempio

- 4 task $T = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.
- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.





Project Scheduling con risorse limitate

5.5. RCPSP(C_{max}): Altri lower bound (continua)

- *Calcolo di* δ_1: risorsa R_1: $r_{i1} = (3, 2, 0, 2)$, $Q_1 = 3$

  Risolviamo la seguente istanza di $P3|r_i, \text{size}, \text{prec}|L_{max}$

task	T_1	T_2	T_4
p_i	7	6	5
r_i	0	7	7
d_i	7	13	15

  con $T_1 < T_2, T_4$.

  La schedula ottima è $t_1 = 0$, $t_2 = 7$, $t_4 = 13$ con $L^*_{max} = 3$.

  Pertanto $\delta_1 = 3$.

- *Calcolo di* δ_2: risorsa R_2: $r_{i2} = (2, 0, 1, 2)$, $Q_2 = 3$

  Risolviamo la seguente istanza di $P3|r_i, \text{size}, \text{prec}|L_{max}$

task	T_1	T_3	T_4
p_i	7	2	5
r_i	0	13	7
d_i	7	15	15

  con $T_1 < T_3, T_4$.

  La schedula ottima è $t_1 = 0$, $t_3 = 13$, $t_4 = 7$ con $L^*_{max} = 0$.

  Pertanto $\delta_2 = 0$.

- Pertanto, $LB3 = LB1 + \max(\delta_1, \delta_2) = 15 + 3 = 18$

Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl.

Euristica shifting bottleneck

- E' una *procedura iterativa* (analoga a quella per il job-shop) che consta di s iterazioni.
- Ad *ogni iterazione* viene individuata la *risorsa* che costituisce il *bottleneck* del progetto, e determinata la *miglior schedula dei task* che ne fanno uso.
- Tale schedula definisce ulteriori precedenze tra i task T, che si traducono nella definizione di archi aggiuntivi a $\vec{G}_0$.
- Indichiamo con $\mathcal{R}$ l'insieme delle s risorse.
- Ad una generica iterazione, indichiamo con $\mathcal{R}_0$ il sottoinsieme delle risorse per le quali è stata specificata la schedula dei task che ne fanno uso.
- Pertanto nella generica iterazione il primo passo da fare è *individuare la risorsa bottleneck* in $\mathcal{R} \setminus \mathcal{R}_0$.

Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

- Per determinare la *risorsa bottleneck*, si individua quale risorsa in $\mathcal{R}\backslash\mathcal{R}_0$ è *causa del maggior accumulo di ritardo* a causa del sequenziamento dei task che ne fanno uso.

- Sia $\vec{G}^+(\mathcal{R}_0)$ il grafo orientato (aciclico) ottenuto da $\vec{G}_0$ aggiungendo gli archi rappresentanti le nuove precedenze dovute alla schedulazione dei task che impiegano le risorse $\mathcal{R}_0$.

- La *lunghezza del cammino critico* di $\vec{G}^+(\mathcal{R}_0)$ è pertanto il *makespan* $C_{max}(\mathcal{R}_0)$ della *schedula ottima* supponendo di *rilassare* i vincoli sulle *altre risorse* $\mathcal{R}\backslash\mathcal{R}_0$.

- La *risorsa bottleneck* è quella k per cui $C_{max}(\mathcal{R}_0 \cup \{R_k\}) = C_{max}(\mathcal{R}_0) + \delta_k$ è massimo tra tutte le risorse R_h, $h = 1, \ldots, s$, dove δ_h è il minimo tempo addizionale necessario per schedulare i task che impiegano la risorsa R_h.

- *Calcolo* di δ_h:
  Per ogni task T_i, siano r_i il suo earliest start time ES_i e d_i il suo latest finish time LF_i calcolato su $\vec{G}^+(\mathcal{R}_0)$, noto $C_{max}(\mathcal{R}_0)$.

Project Scheduling con risorse limitate

5.6. RCPSP($C_{\max}$): Euristica shift. bottl. (continua)

- Determinare δ_h equivale a risolvere il problema $Pm|r_i, \text{ size}, \text{prec}|L_{\max}$, cioè un problema di scheduling con m *macchine parallele* , con *release date*, *due date* e *precedenze*, in cui *ciascun task* T_i *richiede simultaneamente* r_{ih} *macchine* e *l'obiettivo è minimizzare la lateness massima*; $\delta_h = \max [0, L^*_{\max}]$.

- In particolare, le *release date* sono gli ES_i e le *due date* sono i LF_i, le precedenze sono quelle degli archi di $\vec{G}^+(\mathcal{R}_0)$ e $m = Q_h$.

- Determinata la *risorsa bottleneck* R_k, i task vengono schedulati su di essa secondo la schedula ottenuta risolvendo il problema ausiliario $Pm|r_i, \text{ size}, \text{prec}|L_{\max}$.

- Questo consiste nel considerare le nuove precedenze dovute alla schedulazione dei task che usano la risorsa R_k, ovvero definire il nuovo grafo aggiunto $\vec{G}^+(\mathcal{R}_0 \cup \{R_k\})$ ottenuto aggiungendo i relativi archi a $\vec{G}^+(\mathcal{R}_0)$.

- Prima di terminare la generica iterazione si può eventualmente riottimizzare localmente la soluzione parziale ottenuta sequenziando le risorse $\mathcal{R}_0 \cup \{R_k\}$.

Project Scheduling con risorse limitate

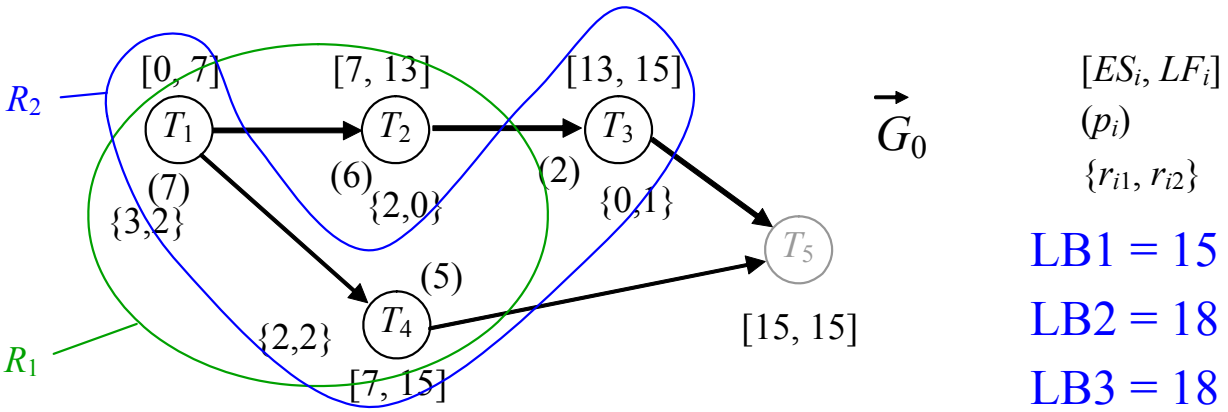
5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

- Riconsideriamo il precedente.

Esempio

- 4 task $T = \{T_1, T_2, T_3, T_4\}$ con durate $p = (7, 6, 2, 5)$ con relazioni di precedenza: $T_1 < T_2, T_4$; $T_2 < T_3$.

- Risorse R_1, R_2, di quantità $Q_1 = 3$ e $Q_2 = 3$. Siano $r_{i1} = (3, 2, 0, 2)$ e $r_{i2} = (2, 0, 1, 2)$ le quantità di R_1, R_2 utilizzate dai task T_i.



Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

- Applichiamo l'euristica *shifting bottleneck*.

Iniz.: $\mathcal{R}_0$ è vuoto.

Iter. 1: $C_{max}(\mathcal{R}_0) = 15$ pari alla lunghezza del cammino critico di $\vec{G}^{+}(\mathcal{R}_0) = \vec{G}_0$.

Determiniamo la risorsa bottleneck

Risorsa R_1: $r_{i1} = (3, 2, 0, 2)$, $Q_1 = 3$

Risolviamo la seguente istanza di $P3|r_i, \text{size}, \text{prec}|L_{max}$

task	T_1	T_2	T_4
p_i	7	6	5
r_i	0	7	7
d_i	7	13	15

con $T_1 < T_2, T_4$.

Schedula ottima: $t_1 = 0, t_2 = 7, t_4 = 13$ con $L^*_{max} = 3$.

Risorsa R_2: $r_{i2} = (2, 0, 1, 2)$, $Q_2 = 3$

Risolviamo la seguente istanza di $P3|r_i, \text{size}, \text{prec}|L_{max}$

task	T_1	T_3	T_4
p_i	7	2	5
r_i	0	13	7
d_i	7	15	15

con $T_1 < T_3, T_4$.

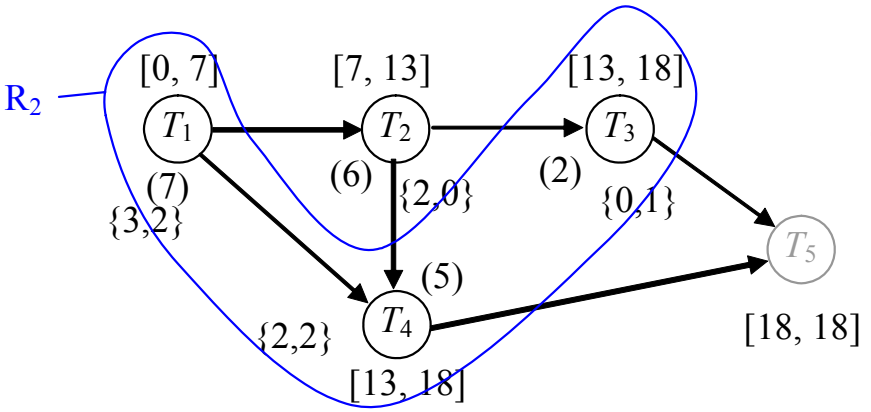
Schedula ottima: $t_1 = 0, t_3 = 13, t_4 = 7$ con $L^*_{max} = 0$.

Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

La risorsa candidata è R_1 ($L^*_{max} = 3$). Sia $\mathcal{R}_0 = \{R_1\}$.

Scheduliamo i task che usano la risorsa R_1 secondo $t_1 = 0$, $t_2 = 7$, $t_4 = 13$, che definisce le nuove precedenze (archi) $T_2 < T_4$. Si ha:



$C_{max}(\mathcal{R}_0) = 18$

Iter. 2: $C_{max}(\mathcal{R}_0) = 18$ pari alla lunghezza del cammino critico di $\vec{G}^+(\mathcal{R}_0) = \vec{G}_0$.

Rimane solo da schedulare rispetto alla risorsa R_2 (che quindi sarà la risorsa bottleneck).

Risorsa R_2: $r_{i2} = (2, 0, 1, 2)$, $Q_2 = 3$

Risolviamo la seguente istanza di $P3|r_i, \text{size}, \text{prec}|L_{max}$

task	T_1	T_3	T_4
p_i	7	2	5
r_i	0	13	13
d_i	7	18	18

con $T_1 < T_3, T_4$.

Schedula ottima: $t_1 = 0, t_3 = 13, t_4 = 13$ con $L^*_{max} = 0$.

Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

Si noti che nessuna nuova precedenza è definita da tale soluzione. Pertanto alla fine della seconda iterazione il grafo esteso $\vec{G}^{+}(\{R_1, R_2\})$ coincide con il grafo $\vec{G}^{+}(\{R_1\})$ ottenuto alla fine della iterazione precedente.

Nel caso specifico infatti il grafo esteso $\vec{G}^{+}(\{R_1\})$ al termine della iterazione 1 è già ammissibile.

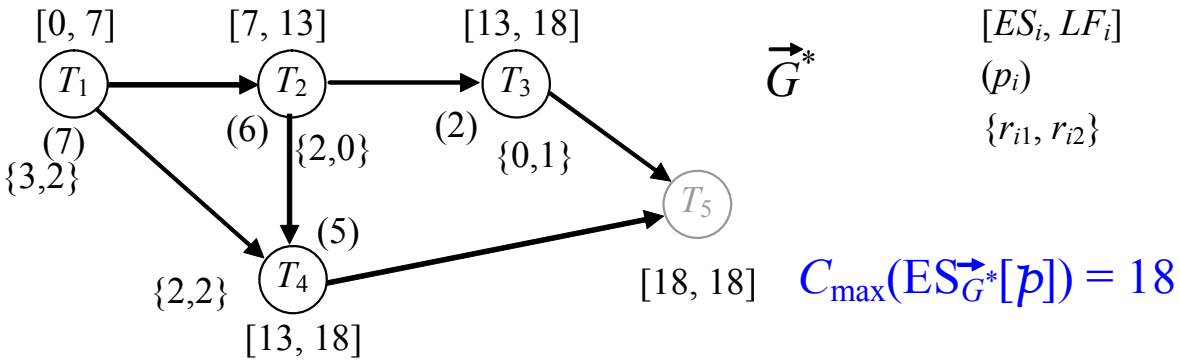
Infatti abbiamo quantità sufficiente per la risorsa R_2. Si noti che $\alpha_{r_2}(\vec{G}_t^{+}(\{R_1\})) \leq \bar{r}_2 = 3$.

- Infine, il grafo esteso è anche ottimo, pertanto $\vec{G}^{*} = \vec{G}^{+}(\{R_1, R_2\})$, perchè $C_{max}(\{R_1, R_2\}) = \text{LB2} = 18$.

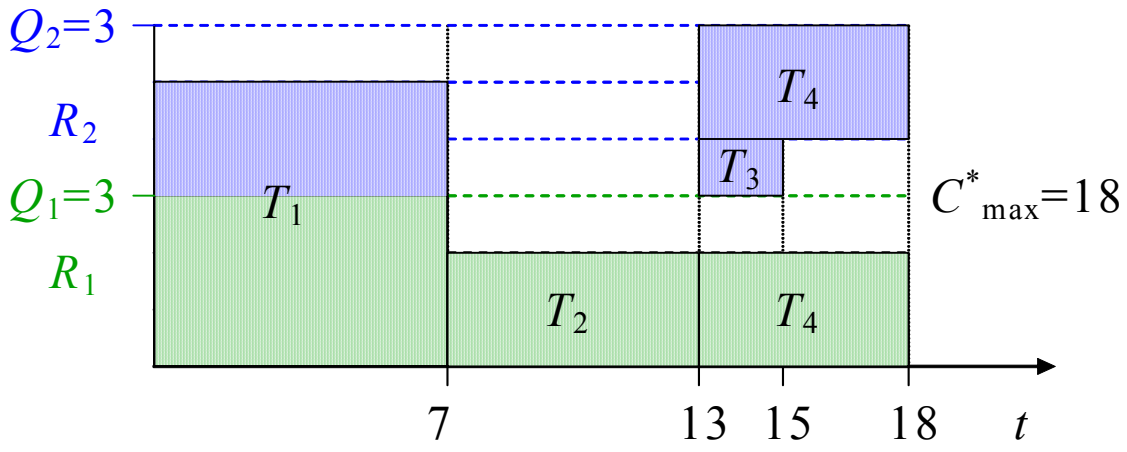
Project Scheduling con risorse limitate

5.6. RCPSP(C_{max}): Euristica shift. bottl. (continua)

- Ridisegnamo $\vec{G}^* = \vec{G}^+(\mathcal{R})$.



- A partire da $\vec{G}^* = \vec{G}^+(\mathcal{R})$ si ottiene la seguente $ES_{\vec{G}^*}[p]$.



- Concludiamo notando che nel caso in cui le risorse rinnovabili siano tutte disponibili per una unità i task che ne fanno uso sono mutuamente in conflitto e tali conflitti possono essere rappresentati con archi disgiuntivi aggiunti a G_0 come nel job-shop.