

Schedulazione delle Attività di un Progetto in Presenza di Multi-Calendarì

Dott.ssa Maria Silvia Pini

Dipartimento di Matematica Pura e Applicata

Email: mpini@math.unipd.it

Resp. accademico: Prof.ssa Francesca Rossi

1 Introduzione

La schedulazione delle attività di un progetto riguarda l'assegnamento di risorse limitate (macchine, denaro, personale) alle attività (fasi di progetto, servizi, lezioni) sull'asse temporale. Più precisamente, determina quando un'attività deve iniziare e quando deve finire, tenendo conto della sua durata, delle attività che la precedono, delle relazioni con i predecessori, della disponibilità di risorse e della data di consegna del progetto. Una descrizione più dettagliata di questa tematica si può trovare in [3,2].

In molte applicazioni reali è necessario tener conto del fatto che alcune risorse (personale o macchine) hanno dei calendari diversi che specificano gli intervalli di tempo in cui esse non sono disponibili [1]. In questo documento presenteremo un algoritmo di schedulazione efficiente in questi scenari che calcola i tempi di inizio e di completamento delle attività che devono essere realizzate il più presto possibile. Altri algoritmi efficienti per risolvere alcune varianti di questo problema, come ad esempio calcolare la schedulazione delle attività fatta il più tardi possibile, vengono illustrati in [1].

2 Il modello

La schedulazione delle attività di un progetto riguarda l'assegnamento di intervalli di tempi di esecuzione alle varie attività. Per definire il modello di questo problema abbiamo bisogno di conoscere i seguenti elementi:

- l'insieme delle attività, che chiamiamo $V = \{0, 1, \dots, n, n + 1\}$;

- per ogni attività $i \in V$, il numero intero $p_i \geq 0$ che indica la durata dell'attività i (dove le attività 0 e $n + 1$ con durate $p_0 = p_{n+1} = 0$ sono attività fittizie che rappresentano l'inizio e la fine del progetto).

Inoltre, per ogni coppia di attività i e j può essere indicato:

- il *minimo scarto temporale* d_{ij}^{min} , che ci dice che l'attività j può essere iniziata d_{ij}^{min} unità di tempo dopo l'inizio dell'attività i al più presto;
- il *massimo scarto temporale* d_{ij}^{max} , che ci dice che l'attività j deve essere iniziata d_{ij}^{max} unità di tempo dopo l'inizio dell'attività i al più tardi.

Le attività e gli scarti temporali si possono rappresentare con una rete N definita da un insieme di nodi V e un insieme di archi E dove:

- l'insieme dei nodi coincide con l'insieme delle attività V ;
- l'insieme degli archi E è definito nel modo seguente:
 - per ogni minimo scarto temporale d_{ij}^{min} , introduciamo un arco pesato (i, j) dal nodo i al nodo j con peso $\delta_{ij} = d_{ij}^{min}$;
 - per ogni massimo scarto temporale d_{ij}^{max} , introduciamo un arco pesato all'indietro (j, i) dal nodo j al nodo i con peso $\delta_{ji} = -d_{ij}^{max}$.

Per ogni nodo i della rete N , indicheremo con

- $Succ(i)$ l'insieme di tutti i successori di i in N ;
- $Prec(i)$ l'insieme di tutti i predecessori di i in N .

Quando dobbiamo schedare progetti reali dobbiamo tenere in considerazione le interruzioni come i week-end e le vacanze in cui alcuni lavoratori e alcune macchine non sono disponibili [4]. Si parla in questo ambito di *break calendars* (cioè di calendari interrotti). In queste situazioni dobbiamo distinguere le attività in:

- *attività interrompibili*, cioè attività che si possono interrompere. Chiameremo V^{bi} l'insieme delle attività (break-)interrompibili. Per ogni attività interrompibile i , viene indicato un minimo tempo di esecuzione ϵ_i . In applicazioni pratiche generalmente si assume $\epsilon_i = 1$.
- *attività non interrompibili*, cioè attività che non si possono interrompere. Pertanto per ognuna di queste attività i si assume che il minimo tempo di esecuzione coincide con la sua durata, cioè $\epsilon_i = p_i$. Chiameremo V^{ni} l'insieme delle attività non interrompibili.

Un *break-calendar* (o più brevemente calendario) è una funzione

$$b : \mathcal{R}_{\geq 0} \rightarrow \{0, 1\}$$

che mappa i numeri reali positivi nell'insieme contenente solo 0 e 1 e che ha la seguente definizione:

- $b(t) = 1$ indica che il tempo t appartiene a un periodo lavorativo;
- $b(t) = 0$ indica che il tempo t appartiene ad un periodo di interruzione.

Data una funzione di calendario b , per $0 \leq \alpha < \beta$, il *tempo totale di esecuzione* nell'intervallo $[\alpha, \beta[$ è dato dal seguente integrale $\int_{\alpha}^{\beta} b(\tau) d\tau$ che coincide con l'area totale sottostante al grafico della funzione b nell'intervallo che va da α a β .

Example 1. La Figura 1 mostra un esempio di una funzione di calendario con il corrispondente tempo totale di esecuzione fino al tempo t . Questo tempo di esecuzione è dato dall'integrale $\int_0^t b(\tau) d\tau$ che è una funzione lineare e continua in t i cui punti d'angolo coincidono con l'inizio e la fine delle interruzioni nel calendario b . In questo esempio si può vedere che l'intervallo $[0, 5[$ e l'intervallo $[10, 15[$ sono due intervalli di lavoro (perchè $b(t) = 1$ in questi intervalli), mentre l'intervallo $[5, 10[$ è un intervallo di interruzione dell'attività lavorativa (perchè $b(t) = 0$ in questo intervallo). Il tempo totale di esecuzione:

- coincide con t finchè t è nell'intervallo di lavoro $[0, 5[$ (quindi, per esempio, è uguale a 1 quando $t=1$, è uguale a 5 quando t è uguale a 5 e così via);
- continua ad essere uguale a 5 in tutto il periodo di interruzione $[5, 10[$;
- coincide con $t - 5$ quando t è nell'intervallo di lavoro $[10, 15[$.

Quindi il tempo totale di esecuzione da $t = 0$ a $t = 15$, che è dato dall'intergrale $\int_0^{15} b(\tau) d\tau$, è 10. □

Per realizzare le attività $i = 1, \dots, n$ del progetto occorrono delle risorse rinnovabili che possono essere macchine o lavoratori. In questo documento supporremo che la capacità delle risorse rinnovabili sia non limitata¹.

Nella pratica diverse risorse hanno diversi calendari. Data un'attività i otteniamo la corrispondente *attività di calendario* b_i settando:

¹ Il caso in cui la capacità delle risorse è limitata viene trattato nella Sezione 4 di [1].

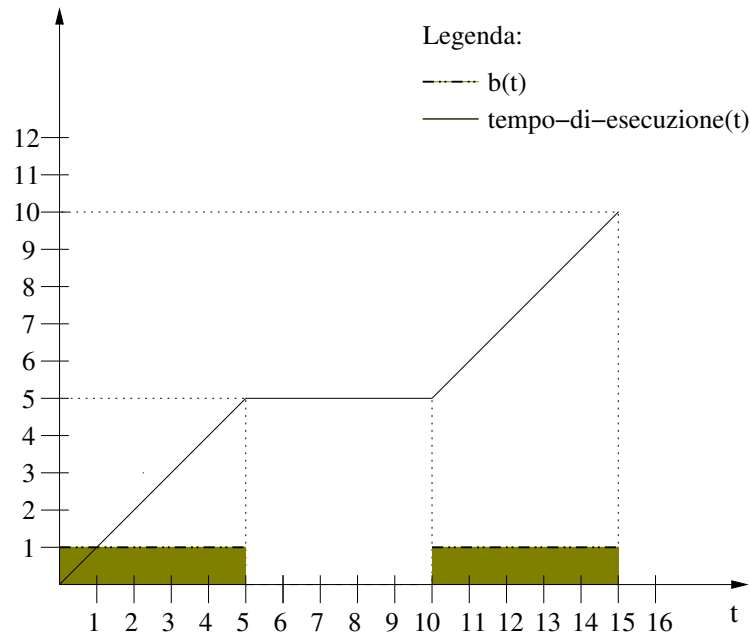


Figura1. Il calendario b e il tempo totale di lavoro.

- $b_i(t) = 0$, se c'è una risorsa usata dall'attività i che non è disponibile al tempo t a causa di un periodo di interruzione;
- $b_i(t) = 1$, se tutte le risorse usate dall'attività i sono disponibili al tempo t .

Nel seguito imporreemo queste restrizioni che sono generalmente accettate in pratica per le attività interrompibili:

1. ogni attività non può essere interrotta all'interno di un intervallo di lavoro;
2. ogni attività interrotta deve ripartire nel primo istante di tempo del prossimo intervallo di lavoro (cioè al tempo $t' = \min\{\tau > t \mid b_i(\tau) = 1\}$);
3. la minima lunghezza di un intervallo di lavoro tra due periodi di interruzione successivi nel calendario b_i dell'attività i deve essere maggior o uguale al minimo intervallo di esecuzione ϵ_i (cioè $b_i(\tau) = 1, \forall \tau \in [S_i, S_i + \epsilon_i[$, dove S_i è il tempo di inizio dell'attività i).

Date le restrizioni 1 e 2, il tempo di completamento dell'attività i è univocamente determinato da $C(S_i) = \min\{t \geq S_i + p_i \mid \int_{S_i}^t b_i(\tau) d\tau = p_i\}$. Notare che $C(S_i) \geq S_i + p_i$ per le attività interrompibili i , mentre $C(S_i) = S_i + p_i$ per le attività non interrompibili.

3 Un algoritmo di schedulazione

Presenteremo ora un algoritmo efficiente capace di calcolare la schedulazione delle attività al più presto nello scenario che abbiamo illustrato nella sezione precedente [1].

L'algoritmo (Algoritmo 1) prende in input l'insieme V delle attività, le funzioni di calendario b_1, \dots, b_n delle varie attività e gli scarti temporali δ_{ij} , per ogni coppia di attività i e j in V e ritorna in output il vettore $ES = (ES_0, \dots, ES_{n+1})$ contenente i tempi di inizio delle varie attività. Più precisamente, nel posto 0 del vettore ES c'è il tempo di inizio dell'attività 0, nel posto 1 del vettore ES c'è il tempo di inizio dell'attività 1, e così via.

Inizialmente settiamo il vettore ES con $(0, -\infty, \dots, -\infty)$ e successivamente ritardiamo le attività finché tutti i vincoli di calendario sono soddisfatti.

Consideriamo poi una coda Q che contiene tutte le attività in cui è stato determinato un tempo di inizio anticipato. In ogni iterazione, eliminiamo un attività i dalla coda Q .

- Prima verifichiamo se il tempo di inizio ES_i rispetta oppure no il calendario b_i . Per far questo, calcoliamo il primo istante di tempo $t^* \geq ES_i$ per cui l'intervallo $[t^*, t^* + \epsilon_i[$ non contiene interruzioni. Se $t^* = \infty$, allora vuol dire che nessuna soluzione è ammissibile e quindi l'algoritmo termina. Nel caso in cui $ES_i < t^*$, ritardiamo il tempo di inizio dell'attività i fino al tempo t^* .
- Poi controlliamo i vincoli temporali per tutti i successori diretti $j \in Succ(i)$ dell'attività i nella rete N . Per far questo calcoliamo il primo istante di tempo $t^* = \min\{t \geq \max\{0, ES_j\} \mid \int_{ES_i}^t b_{ij}(\tau) d\tau \geq \delta_{ij}\}$ dell'attività j dato il tempo di inizio ES_i per l'attività i che la precede. Se $ES_j < t^*$ allora la schedulazione ES non soddisfa i vincoli temporali legati agli scarti temporali e quindi ritardiamo ES_j fino al tempo t^* . Poi, se j non appartiene alla coda Q , lo inseriamo in Q . Quest'ultima operazione la effettuiamo anche se $b_j(\tau) = 0$ per qualche $\tau \in [t^*, t^* + \epsilon_j[$.

Denotiamo con B il numero delle interruzioni in tutti i calendari delle attività, cioè $B = \sum_{i \in V} B_i$. Se c'è una schedulazione ES ammissibile che soddisfa sia i vincoli di calendario che i vincoli temporali, l'Algoritmo 1 la restituisce dopo aver visitato ogni arco (i, j) al massimo $|V|(B+1)$ volte, dove ognuno delle B interruzioni è considerata solo una volta. Se i calendari sono dati come liste ordinate di tempi di inizio e tempo di fine delle interruzioni, la complessità dell'Algoritmo 1 è $\mathcal{O}(|V||E|(B+1))$. Se dopo $|V||E|(B+1)$ iterazioni la coda Q contiene ancora delle attività, allora vuol dire che nessuna schedulazione è ammissibile.

Algorithm 1: Schedulazione al più presto di attività con multi-calendari

input: $V = (0, 1, \dots, n, n + 1)$: l'insieme delle attività
 b_1, \dots, b_n : le funzioni di calendario delle attività $1, \dots, n$
 $\delta_{i,j}$: il minimo scarto temporale tra due attività i e j per ogni coppia di attività i e j
output: $ES = (ES_0, \dots, ES_{n+1})$: vettore con i tempi di inizio delle attività

for $i \in V - \{0\}$ **do**
 $ES_i \leftarrow -\infty$;
 $ES_0 = 0$;
 $Q \leftarrow \{V - \{0\}\}$; ** Q è una coda**

while $Q \neq \emptyset$ **do**
 Elimina i da Q ;
 Determina $t^* := \min\{t \geq ES_i \mid b_i(\tau) = 1, \forall \tau \in [t, t + \epsilon_i[\}$;
 if $t^* = \infty$ **then**
 Termina; **non c'è una soluzione ammissibile**
 else
 if $ES_i < t^*$ **then**
 $ES_i \leftarrow t^*$;
 for $j \in Succ(i)$ **do**
 Determina $t^* := \min\{t \geq \max\{0, ES_j\} \mid \int_{ES_i}^t b_{ij}(\tau) d\tau \geq \delta_{ij}\}$;
 if $ES_j < t^*$ oppure $b_j(\tau) = 0$ per qualche $\tau \in [t, t + \epsilon_j[$ **then**
 if $ES_j < t^*$ **then**
 $ES_j \leftarrow t^*$;
 if $j \notin Q$ **then**
 Metti j nella coda Q ;
 return ES

Example 2. Proviamo a vedere l'esecuzione dell'Algoritmo 1 su un esempio semplice. Supponiamo di avere due attività, chiamiamole 1 e 2, tali che:

- l'attività 1 deve durare 10 unità di tempo;
- l'attività 2 deve durare 5 unità di tempo;
- l'attività 2 deve essere effettuata solo dopo che è finita l'attività 1;

Assumiamo inoltre che le attività 1 e 2 abbiano i seguenti calendari:

- il periodo di lavoro dell'attività 1 è $[0, 5) \cup [10, 15)$, mentre il periodo di interruzione dell'attività 1 è $[5, 10)$ (più precisamente, $b_1(t) = 1$ in $[0, 5) \cup [10, 15)$ e $b_1(t) = 0$ in $[5, 10)$);

- il periodo di lavoro dell'attività 2 è $[7, 12) \cup [18, 25)$, mentre il periodo di interruzione dell'attività 2 è $[12, 18)$ (più precisamente, $b_2(t) = 1$ in $[7, 12) \cup [18, 25)$ and $b_2(t) = 0$ in $[0, 7) \cup [12, 18)$);

Supponiamo per semplicità, come succede in pratica che il minimo tempo di esecuzione per le attività sia 1, cioè $\epsilon_1 = \epsilon_2 = 1$.

L'algoritmo 1 su questo esempio funziona come segue. Inizialmente istanzia il vettore ES con uno zero all'inizio e tutti gli altri valori uguali a $-\infty$. La coda Q nella fase iniziale contiene le attività 1 e 2 e l'attività di fine progetto, chiamiamola 3. Nella prima iterazione eliminiamo l'attività 1 dalla coda Q .

- Calcoliamo il primo istante di tempo $t^* \geq ES_1$ per cui l'intervallo $[t^*, t^* + 1[$ non contiene interruzioni. $t^* = 0$, quindi settiamo ES_1 a zero.
- Poi controlliamo i vincoli temporali per tutti i successori diretti dell'attività 1, che in questo caso sono costituiti solo l'attività 2. Per far questo calcoliamo il primo istante di tempo $t^* = \min\{t \geq \max\{0, ES_2\} \mid \int_{ES_1}^t b_{12}(\tau) d\tau \geq 10\}$ che è $t = 15$. Quindi pongo $ES_2 = 15$. Poi non inserisco l'attività 2 nella coda Q perchè c'è già.

Quindi eseguiamo un'altra iterazione. Eliminiamo l'attività 2 dalla coda Q .

- Calcoliamo il primo istante di tempo $t^* \geq ES_2 = 15$ per cui l'intervallo $[t^*, t^* + 1[$ non contiene interruzioni. $t^* = 18$, quindi settiamo ES_2 a 18.
- Controlliamo i vincoli temporali per tutti i successori diretti dell'attività 2, che in questo caso sono costituiti solo dall'attività 3 di fine progetto. Per far questo calcoliamo il primo istante di tempo $t^* = \min\{t \geq \max\{0, ES_3\} \mid \int_{ES_2}^t b_{23}(\tau) d\tau \geq 5\}$ che è $t = 23$.

Nell'iterazione successiva eliminiamo l'attività 3 e terminiamo l'algoritmo restituendo una schedulazione ammissibile in cui il tempo di inizio dell'attività 1 è l'istante 0 e il tempo di fine è l'istante 15, il tempo di inizio dell'attività 2 è l'istante 18 e il tempo di fine dell'attività 2 è l'istante 23. \square

4 Sommario ed articoli scientifici correlati

In questo documento abbiamo considerato il problema della schedulazione delle attività di un progetto quando le varie attività hanno diversi calendari in cui possono essere effettuate e ci sono dei vincoli temporali di precedenza tra le varie attività. Una descrizione più approfondita del problema classico senza multi-calendari si può trovare in [3,2].

In particolare, abbiamo presentato un algoritmo efficiente per calcolare una schedulazione ammissibile ‘al più presto’ delle attività di un progetto [1]. L’algoritmo si basa su diverse assunzioni che sono comunemente accettate nella pratica. Più precisamente, abbiamo assunto che:

- ogni attività non possa essere interrotta all’interno di un intervallo di lavoro,
- ogni attività interrotta debba ripartire nel primo istante di tempo del prossimo intervallo di lavoro,
- che la minima lunghezza di un intervallo di lavoro tra due periodi di interruzione successivi nel calendario di un’attività debba essere maggior o uguale al minimo tempo di esecuzione, che generalmente nella pratica è uguale a 1.

Un algoritmo che fa le stesse assunzioni, ma che invece di calcolare una schedulazione delle attività al più presto, calcola una schedulazione delle attività ‘al più tardi’ viene descritto nella Sezione 3 di [1].

In questo documento abbiamo considerato solo scenari in cui non ci sono vincoli sulle risorse. Un metodo di soluzione che tiene conto anche di questi vincoli viene presentato nella Sezione 4 di [1]. Questo metodo è un adattamento di una procedura descritta in [2].

La presenza di multi-calendari per le attività nel contesto della schedulazione di progetti è stata considerata anche in [5]. Comunque in questo articolo la differenza dei calendari delle varie attività riguarda solo la granularità dei vari calendari. Per esempio, il calendario di un’attività è espresso in ore, il calendario di un’altra attività è espresso in giorni, mentre i calendari di altre attività sono espressi in settimane. La tecnica risolutiva che viene utilizzata in questo scenario consiste semplicemente nell’esprimere ogni calendario in termini di ore nel nostro esempio, cioè in termini della minima unità di tempo che viene considerata nei vari calendari. In questo modo tutti i calendari vengono espressi nella stessa unità di tempo e quindi si possono usare le tecniche risolutive classiche dei problemi di schedulazione.

Riferimenti bibliografici

1. B. Franck, K. Neumann, and C. Schwindt. Project scheduling with calendars. *OR Spectrum*, 23(3):325–334, 2001.
2. B. Franck, K. Neumann, and C. Schwindt. Truncating branch-and-bound, schedule-construction, and schedule improvement procedures for resource-constrained project scheduling. *OR Spectrum*, 23(3):297–324, 2001.

3. K. Neumann and C. Schwindt. Activity-on-node networks with minimal and maximal time-lags and their application to make-to-order production. *OR Spectrum*, 19:205–217, 1997.
4. C. Schwindt and N. Trautmann. Batch scheduling in process industries: An application of resource-constrained project scheduling. *OR Spectrum*, 22:501–524, 200.
5. S. Spranger and F. Bry. Multi-calendar appointment scheduling: calendar modeling and constraint reasoning. In *Proceedings of PATAT 2006 International Conference on the Practice and Theory of Automated Timetabling*, pages 496–501, 2006.