Bollettino
Unione Matematica Italiana

Sezione A – La Matematica nella Società e nella Cultura

Antonio Iovanella

Algoritmi on-line per lo scheduling di attività multiprocessore

Bollettino dell'Unione Matematica Italiana, Serie 8, Vol. **7-A**—*La Matematica nella Società e nella Cultura* (2004), n.3, p. 531–534.

Unione Mastematica Italiana

<http://www.bdim.eu/item?id=BUMI_2004_8_7A_3_531_0>

Bollettino dell'Unione Matematica Italiana, Unione Mastematica Italiana, 2004.

Bollettino U. M. I.
La Matematica nella Società e nella Cultura
Serie VIII, Vol. VII-A, Dicembre 2004, 531-534

Algoritmi on-line per lo scheduling di attività multiprocessore.

Antonio Iovanella

1. – Introduzione.

La teoria dello scheduling affronta un particolare problema di ordinamento riguardante l'allocazione di risorse nel tempo per eseguire un insieme di attività facenti parte di un certo processo. Le attività competono per le risorse, le quali possono essere di diversa natura come ad esempio processori e strumenti, e sono caratterizzate da parametri quali il tempo di rilascio, il tempo di processamento, il tempo di consegna ed altre funzioni che descrivono le relazioni con le risorse disponibili. Ciò posto, possono essere definiti diversi criteri al fine di misurare la qualità delle prestazioni, quali ad esempio il tempo di completamento totale delle attività o il loro tempo medio di permanenza nel sistema.

Nella teoria tradizionale dello scheduling la letteratura si è concentrata principalmente su problemi nei quali ogni attività è processata soltanto da un singolo processore per volta. Invece negli ultimi tempi sono emersi nuovi approcci teorici per modellare problemi di scheduling su architetture parallele e, tra questi, il *multiprocessor task scheduling*, ovvero il sequenziamento di attività multiprocessore, oggetto della tesi di dottorato.

In particolare è stata presa in considerazione la seguente classe di problemi di multiprocessor task scheduling: sia dato un insieme $T = \{1, 2, \dots, n\}$ di n attività ed un insieme $M = \{1, 2, \dots, m\}$ di m macchine sulle quali eseguirle; ad ogni task sia associato un tempo di processamento p_j, un tempo di rilascio r_j ed un insieme di processori distinti $fix_j \subseteq M$ ed inoltre le attività una volta iniziate non possano essere interrotte. Ogni processore può eseguire al più una sola attività j alla volta ed ogni attività deve essere processata simultaneamente su ogni macchina dell'insieme fix_j. La funzione obiettivo considerata è quella di minimizzare il tempo di completamento $C_{\max} = \max_{j \in T} C_j$, dove C_j indica il tempo di completamento dell'attività j. Dal punto di vista della complessità computazionale questo problema è stato dimostrato essere NP-Hard [3].

Nei problemi di scheduling viene di solito considerata l'ipotesi semplificativa che il sistema, ovvero lo schedulatore, abbia informazione completa sull'istanza del problema. Per caratterizzare questa classe di problemi viene usato il termine *off-line*, in contrapposizione con quella nella quale i dati delle attività non sono completamente noti fino a che essi non arrivano nel sistema e per la quale viene introdotto il termine *on-line*. La nozione di algoritmo on-line è intesa appunto a formalizzare lo scenario realistico nel quale l'algoritmo non ha accesso all'intera istanza in ingresso, come avviene invece per gli algoritmi off-line, ma la apprende con lo scorrere del tempo e dovendo reagire alle nuove richieste con la sola parziale conoscenza di quanto è già giunto nel sistema.

La classificazione più importante che può essere fatta per gli algoritmi di scheduling on-line è su quale porzione dell'input viene considerata on-line [2]. Tra le diverse possibilità sono stati considerati due diversi paradigmi, ovvero quello in cui le attività possono arrivare *over list* e quello in cui arrivano *over time*. Nel primo caso, le attività j in un insieme T sono ordinate secondo una certa lista e lo schedulatore le considera nell'ordine di tale lista; nel secondo caso invece le attività j arrivano secondo un certo tempo di rilascio r_j e lo schedulatore, che ha conoscenza solo delle attività già rilasciate, le può considerare solo in istanti di tempo $t \geqslant r_j$.

Mentre nella letteratura scientifica i problemi off-line di multiprocessor task scheduling sono stati largamente studiati ed una ampia collezione di risultati è raccolta in [1], la versione on-line è stata per la prima volta trattata in questa sede.

Per indicare le diverse varianti studiate del problema considerato si è usata la notazione standard dello scheduling [1], per cui $P|fix_j,\ r_j\,|C_{\max}$ indicherà il problema nel caso off-line, $P|$ on-line, $fix_j,\ p_j = 1\,|C_{\max}$ indicherà la versione online del problema con tempo di processamento delle attività tutte unitarie ed arrivi over list, $P|$ on-line, $fix_j,\ r_j\,|C_{\max}$ la versione on-line con tempi di processamento arbitrari ed arrivi over time.

I problemi indicati sono stati analizzati dal punto di vista dell'approssimabilità garantita per il caso off-line e dal punto di vista dell'analisi competitiva per i casi on-line. Un algoritmo polinomiale $\mathcal{A}$ è una σ-approssimazione per il valore del tempo di completamento $C_{\max}$ di un problema di scheduling se per ogni istanza $\mathcal{I}$ del problema con soluzione ottima $C_{\max}^{\mathcal{OPT}}(\mathcal{I})$ si ha che:

$$\sigma \leqslant \frac{C_{\max}^{\mathcal{A}}(\mathcal{I})}{C_{\max}^{\mathcal{OPT}}(\mathcal{I})} \tag{1}$$

Naturalmente più tende ad uno il rapporto di approssimazione $\sigma \geqslant 1$, migliore sarà l'algoritmo.

L'analisi competitiva è anch'essa una analisi del caso peggiore, ma riferita ad algoritmi on-line; infatti, un algoritmo on-line $\mathcal{A}$ è detto c-competitivo se esiste una costante α tale che per ogni istanza di input $\mathcal{I}$ si ha che:

$$C_{\max}^{\mathcal{A}}(\mathcal{I}) \leqslant c \cdot C_{\max}^{\mathcal{OPT}}(\mathcal{I}) + \alpha \tag{2}$$

L'analisi competitiva è effettuata generalmente introducendo un algoritmo avversario deterministico che, conoscendo la sequenza in anticipo, è in grado di ottenere la soluzione ottima off-line.

2. – Analisi competitiva degli algoritmi on-line.

Dato il problema di multiprocessor task scheduling introdotto nella sezione precedente, una schedula ammissibile è una assegnazione di intervalli di tempo $[S_j,\ S_j + p_j)$, $S_j \geqslant r_j$ alle attività $j \in T$ tale che ogni processore $i \in M$ esegue una attività alla volta ed ogni attività è eseguita simultaneamente dai processori in fix_j. S_j è l'istante di inizio e $C_j = S_j + p_j$ il tempo di completamento dell'attività.

2.1. *Tempi di processamento unitari.*

In questo prima parte è stato analizzato il caso in cui $p_j = 1$, $\forall j \in T$ ed inoltre è stata posta l'ipotesi che le attività possano essere processate al più su $k > 0$ macchine. L'istanza del problema è stata rappresentata mediante un grafo di incompatibilità, ovvero un grafo non orientato $G(V, E)$ nel quale V è l'insieme delle attività ed un arco $(i, j) \in E$ esiste se e solo se le due attività hanno un processore in comune, quindi se $fix_i \cap fix_j \neq \emptyset$.

Per risolvere questo problema si è introdotto inizialmente un algoritmo chiamato *FFS* nel quale le attività j, considerate in ordine arbitrario, si assegnano al primo intervallo di tempo $[t, t+1)$, $t \geq r_j$ nel quale i processori in fix_j sono liberi. Utilizzando dei risultati preliminari di colorazione su grafi k-upli si è ottenuto che:

TEOREMA 1. – *Se* $|fix_j| \leq k$, $\forall j \in T$ *allora l'algoritmo FFS è k-competitivo per* $P|$ *on-line,* fix_j, $p_j = 1 | C_{\max}$ *ed è* $(k+1)$*-competitivo per* $P|$ *on-line,* fix_j, $p_j = 1$, $r_j | C_{\max}$.

Chiaramente questo risultato mostra prestazioni molto negative per $k = m$, per cui è stato migliorato considerando un algoritmo, chiamato $FFS+$, che esegue una partizione dell'insieme delle attività in due sottoinsiemi, T^- e T^+, assegnando rispettivamente a T^- le attività con $|fix_j| \leq \sqrt{m}$ ed a T^+ quelli con $|fix_j| > \sqrt{m}$; le due partizioni vengono poi schedulate una dopo l'altra tramite l'algoritmo *FFS*. In questo caso si è dimostrato che:

TEOREMA 2. – *L'algoritmo* $FFS+$ *è* $2\sqrt{m}$*-competitivo per* $P|$ *on-line,* fix_j, $p_j = 1 | C_{\max}$ *ed è* $2(\sqrt{m}+1)$*-competitivo per* $P|$ *on-line,* fix_j, $p_j = 1$, $r_j | C_{\max}$.

2.2. *Tempi di processamento arbitrari.*

Nel caso più generale di tempi di processamento arbitrari p_j, le tecniche sopra descritte sono state generalizzate introducendo un algoritmo chiamato *GFFS* nel quale, considerando le attività j in ordine arbitrario, si assegnano le stesse al primo intervallo di tempo $[t, t+p_j)$, $t \geq r_j$ nel quale i processori in fix_j sono liberi. Per questo algoritmo si può dimostrare che:

TEOREMA 3. – *L'algoritmo GFFS è m-competitivo per* $P|$ *on-line,* $fix_j | C_{\max}$ *ed è* $(m+1)$*-competitivo per* $P|$ *on-line,* fix_j, $r_j | C_{\max}$.

Anche in questo caso i primi risultati presentati possono essere migliorati. Al tal fine è stato prima studiato un algoritmo off-line che è stato poi esteso al caso on-line. Per il problema off-line $P|fix_j | C_{\max}$ è stato introdotto un algoritmo chiamato *SRS* nel quale, dopo una partizione dell'insieme delle attività in due sottoinsiemi T^- e T^+ così come visto per l'algoritmo $FFS+$, si è proceduto in due modi diversi. Infatti, per le attività nella partizione T^- si è applicato direttamente l'algoritmo *GFFS* mentre in T^+ si è effettuato un arrotondamento del tempo di processamento p_j alla più piccola potenza di due, ovvero si è considerato per ogni attività $j \in T^+$ un nuovo tempo di processamento p_j' tale che $p_j' = 2^a \geq p_j$ e $2^{a-1} < p_j$. Dopo questa fase di

arrotondamento, l'algoritmo procede applicando *GFFS* relativamente alle attività $j \in T^+$. Per l'algoritmo *SRS* si è dimostrato che:

TEOREMA 4. – *SRS è un algoritmo* $3\sqrt{m}$*-approssimato per il problema* $P|fix_j|C_{\max}$.

L'algoritmo on-line *SRS* + funziona considerando due code, una definita attiva e l'altra passiva. Le attività vengono prima collezionate nella lista attiva e schedulate secondo l'algoritmo *SRS*, mentre le attività che nel frattempo arrivano vengono poste nella coda passiva. Quando terminano le attività nella coda attiva, l'algoritmo scambia la lista attiva in passiva e viceversa, continuando questa procedura fino al completamento di tutte le attività. Quindi:

TEOREMA 4. – *SRS* + *è un algoritmo* $3\sqrt{m}$*-competitivo per il problema* $P|$ *online,* fix_j, $r_j|C_{\max}$.

Gli algoritmi esposti in questa sezione sono stati tutti implementati e l'analisi computazionale sul tempo di completamento ha evidenziato risultati molto positivi, sia in termini di gap tra i valori ottenuti e quelli teorici, sia in termini di confronto con gli algoritmi classici di lower bound, quali il calcolo della massima clique e della massima clique pesata calcolata sul grafo di incompatibilità.

3. – Conclusioni.

In questa tesi di dottorato è stato affrontato un particolare problema di scheduling di attività multiprocessore. Lo studio è stato motivato dallo sviluppo di sistemi sempre più complessi, come per esempio le telecomunicazioni, dove si è reso necessario lo studio di algoritmi veloci in grado di adeguarsi alle richieste attraverso prestazioni garantite. A tal fine sono state trattate diverse varianti del problema, sia off-line che on-line, considerando sia tempi di processamento unitari che arbitrari e fornendo algoritmi con prestazione garantita.

BIBLIOGRAFIA

[1] BŁAŻEWICZ J. K., ECKER K. H., PESCH E., SCHMIDT G. e WĘGLARZ J., *Scheduling computer and manufacturing processes*, Springer-Verlag (2001).

[2] FIAT A. e WOEGINGER G. (EDS.), *Online algorithms: the state of the art*, Lectures Notes in Computer Science, Springer-Verlag, **1142** (1998).

[3] HOOGEVEEN J. A., VAN DE VELDE S. L. e VELTMAN B., *Complexity of scheduling multiprocessor tasks with prespecified allocations*, Discrete Applied Mathematics, **55** (1994), 259-272.

Dipartimento di Informatica, Sistemi e Produzione
Università degli Studi di Roma «Tor Vergata»; e-mail: iovanella@disp.uniroma2.it
Dottorato in Ricerca Operativa
(sede amministrativa: Università di Roma La Sapienza) - Ciclo XV
Direttore di ricerca: Prof. G. Di Pillo, Università degli Studi di Roma La Sapienza