UN ALGORITMO DI SIMULATED ANNEALING PER LA SOLUZIONE DI UN PROBLEMA DI SCHEDULING

Crescenzio Gallo[(*)]

SOMMARIO

Nell'articolo viene presentato un algoritmo che sfrutti la tecnica euristica del Simulated Annealing per la soluzione di un problema di scheduling. Il Simulated Annealing è un'approccio nato in analogia con il comportamento dei sistemi fisici durante il processo di raffredamento. Sono illustrate le problematiche relative alla schedulazione, il metodo approssimato di Simulated Annealing e i suoi parametri fondamentali (congelamento, temperatura, raffreddamento, numero di intorni da esplorare), e le scelte operate nell'individuazione di tali parametri per la generazione di un buon algoritmo che risolva in modo efficiente il problema della schedulazione.

ABSTRACT

An algorithm of "Simulated Annealing" for solving scheduling problems is presented. The issues related to scheduling are discussed, together with the Simulated Annealing approximation method and its main parameters (freezing, temperature, cooling, number of neighbourhoods to explore), the choices made in defining them for the generation of a good algorithm that efficiently resolves the scheduling problem.

Keywords: scheduling, simulated annealing.

(*) Dipartimento di Scienze Economiche, Matematiche e Statistiche – Università di Foggia – Via IV Novembre, 1 – 71100 Foggia (Italy) - e-mail: c.gallo@unifg.it

1. INTRODUZIONE

Per la gestione efficiente di un sistema di elaborazione è fondamentale disporre di un algoritmo che specifica la sequenza secondo la quale viene assegnato il servizio di elaborazione ad uno specifico job. Questo algoritmo è chiamato "schedulatore" ed implementa delle metodologie che possono risultare molto onerose in tempo di calcolo. Potrebbe, ad esempio, risultare più costoso elaborare il job "schedulatore" che tutta la sequenza da processare.

E' quindi importante indicare una funzione costo con la quale si può misurare il costo di ogni possibile sequenza, noi vogliamo trovare un ordine di sequenza tale che consenta la minimizzazione della nostra funzione costo. E' quindi chiaro come i problemi di schedulazione sono intimamente connessi alle decisioni di che cosa deve essere fatto e come deve essere fatto.

Dato un problema, diremo di poter estrarre le domande di sequenza associate a tale problema quando sono soddisfatte le seguenti ipotesi:

1. i lavori che devono essere eseguiti sono completamente noti, e possono anche non essere annunciati al processo di schedulazione simultaneamente ma sequenzialmente nel tempo;
2. le risorse che possono essere utilizzate nell'esecuzione dei lavori sono completamente specificate;
3. la sequenza di attività fondamentali richiesta per eseguire ciascuno dei lavori è nota.

1.1. Modelli matematici

Indichiamo con il termine *tempo di completamento* un valore che dipende dalla particolare sequenza di schedulazione. Se un job è nella posizione *i* nella sequenza, il suo tempo di completamento è dato dal tempo di completamento del job che si trova nella posizione *i-1* maggiorato del suo tempo di elaborazione. Indicheremo inoltre con il termine *tempo totale di completamento* la somma dei tempi di completamento dei singoli job.

I problemi di sequenzalizzazione che tratteremo sono detti di tipo *non-preemptive*: il job che è in stato di elaborazione non può essere interrotto per alcuna ragione.

Per la soluzione di problemi di schedulazione descriviamo due modelli matematici che lo formalizzano. Il primo modello formula il problema di scheduling di n jobs su m macchine che minimizzi il tempo totale di completamento. Indichiamo con x_{ij}^k la variabile che vale 1 se il job *j* è il *k*-esimo job eseguito sulla macchina *i*, 0 altrimenti. Indichiamo inoltre con p_{ij} il processing time del processo *j* sulla macchina *i*; rappresenteremo quindi con il simbolo $p_j = \sum_{i=1}^{m} p_{ij}$ il processing time del processo j su tutte le macchine. Il modello è quindi il seguente:

$$\textit{Minimizza} \quad \sum_{k=1}^{n}\sum_{j=1}^{n} p_j x_{ij}^k \qquad i = 1,...,m$$

soggetta ai vincoli :

$$\sum_{k=1}^{n}\sum_{i=1}^{m} x_{ij}^k = 1 \qquad j = 1,...,n$$

$$\sum_{j=1}^{n} x_{ij}^k \le 1 \qquad k = 1,...,n \qquad i = 1,...,m$$

$$x_{ij}^k \in \{0,1\} \qquad i = 1,...,m \qquad j,k = 1,...,n$$

dove il primo vincolo vincola il job *j*-esimo a dover essere eseguito come *k*-esimo sulla macchina *i*-esima per qualche *i* e per qualche *j*.

Il secondo modello che guardiamo è per la schedulazione di *n* jobs su *m* macchine che minimizzi la somma dei tempi di completamento singolo. Indicando con $C_j^k = \sum_{l=1}^{k}\sum_{j=1}^{n} p_j x_{ij}^l$ il tempo di completamento del job *j* schedulato come *k*-esimo sulla macchina *i*. Il relativo modello matematico è quindi:

$$\textit{Minimizzare} \quad \sum_{k=1}^{n}\sum_{i=1}^{m} C_j^k$$

soggetta ai vincoli:

$$\sum_{k=1}^{n}\sum_{i=1}^{m} x_{ij}^k = 1 \qquad j = 1,...,n$$

$$\sum_{j=1}^{n} x_{ij}^{k} \le 1 \qquad k = 1,...,n \qquad i = 1,...,m$$

$$x_{ij}^{k} \in \{0,1\} \qquad i = 1,...,m \qquad j,k = 1,...,n$$

1.2. Approcci di soluzione

Il problema della schedulazione può essere risolto in diversi modi. Un primo modo consiste nell'enumerare tutte le possibili sequenze, calcolando il valore della funzione obiettivo, per scegliere in output la sequenza cui corrisponde il minor valore della funzione obiettivo. Ad esempio se abbiamo 10 jobs su 4 macchine dobbiamo calcolare il valore della funzione obiettivo per ciascuna delle 10! permutazioni possibili. Tale metodo diviene quindi improponibile per problemi di scheduling di media e grande taglia.

Altri metodi classici che risolvono il problema in modo esatto sono costituiti da algoritmi di branch-and-bound, branch-and-bound troncato, programmazione dinamica.

2. IL SIMULATED ANNEALING

Il "simulated annealing" è un approccio basato su concetti di meccanica statistica, ed è motivato da un'analogia con il comportamento dei sistemi fisici durante il processo di raffreddamento.

Tale analogia è meglio illustrata in termini della fisica della formazione dei cristalli. Per formare un cristallo si parte da materiali grezzi allo stato fuso. La temperatura di questo "cristallo fuso" è allora ridotta finchè la struttura del cristallo è "congelata". Se il raffreddamento è eseguito molto rapidamente, si verificano dei fenomeni non desiderabili. In particolare, vengono racchiuse nella struttura del cristallo irregolarità molto estese ed il livello di energia inglobato è molto più alto rispetto a quello che vi sarebbe in un cristallo perfettamente saturato.

Questo processo di "raffreddamento rapido" può essere visto come analogo all'ottimizzazione locale (fig. 1). Gli stati del sistema fisico corrispondono alle soluzioni di un problema di ottimizzazione combinatoriale; l'energia di uno stato corrisponde al costo di una soluzione e la minima energia, o "stato fondamentale" corrisponde ad una soluzione ottima.

SISTEMA FISICO	PROBLEMA DI OTTIMIZZAZIONE
Stato	Soluzione ammissibile
Energia	Costo
Stato fondamentale	Soluzione ottima
Raffreddamento rapido	Ricerca locale
Annealing accurato	Simulated annealing

Fig.1 - L'analogia.

Quando la temperatura è, teoricamente, allo zero assoluto nessuna transizione di stato può portare verso uno stato a più alta energia. Quindi, come nell'ottimizzazione locale, sono proibiti movimenti in salita e le conseguenze di ciò possono essere indesiderabili.

Quando inizia la formazione dei cristalli, il rischio di ottimi locali scadenti è evitato abbassando la temperatura in modo molto graduale, con un processo chiamato di "annealing accurato". In questo processo la temperatura scende molto lentamente attraverso una serie di livelli, ciascuno mantenuto abbastanza a lungo in modo da permettere la ricerca dell' "equilibrio", a quella temperatura, per il cristallo.

Finchè la temperatura è diversa da zero restano sempre possibili movimenti in salita. Evitando che la temperatura si discosti da quella compatibile con il livello energetico dell'equilibrio corrente, possiamo sperare di evitare ottimi locali finchè non giungiamo relativamente vicini allo stato fondamentale.

Il simulated annealing è la controparte algoritmica di questo processo di annealing fisico. Il nome "simulated annealing" si riferisce alla tecnica di simulazione del processo di annealing fisico in congiunzione con un "annealing schedule" di decremento della temperatura; esso può essere visto come un'estensione della tecnica di ottimizzazione locale, in cui la soluzione iniziale viene ripetutamente migliorata tramite piccole perturbazioni locali fino a quando nessuna di tali perturbazioni migliora la soluzione. Il metodo randomizza tale procedura in modo da permettere occasionalmente dei movimenti in salita, cioè delle perturbazioni che peggiorano la soluzione e questo nel tentativo di ridurre la probabilità di bloccarsi in una soluzione localmente ottima ma globalmente scadente.

Il Simulated Annelaing è stato proposto da Metropolis *et al* [Met53] e utilizzato nel campo della fisica statistica allo scopo di determinare le proprietà delle leghe metalliche ad una data temperatura. La possibilità di risolvere problemi di ottimizzazione combinatoriale furono sono state dimostrate indipendentemente da Kirkpartrick *et al* [Kir83] e Cerny [Cer85]. Sono poi state eseguite con notevole successo ulteriori applicazioni riportate in Vecchi e Kirkpatrick [Vec83], Lundy [Lun85], Aerts *et al* [Aer85], Kirkpatrick [Kir84], Heynderickx *at al* [Hey86]

Un semplice schema di simulated annealing è il seguente:

```
1.    Considera una soluzione iniziale S.
2.    Finchè non si ha un congelamento:
      2.1   Per w := 0 sino al numero di intorni considerato:
            2.1.1. Sia S' un vicino di S non esaminato
            2.1.2. Se Costo(S') < Costo(S), poni  S:=S'
            2.1.3.       altrimenti poni S:=S' con una
                                 determinata probabilità.
      2.2   Raffredda la temperatura.
3.    Ritorna S.
```

Fig. 2 - Lo schema del simulated annealing.

Da quanto detto risulta quindi evidente che i parametri fondamentali per la fase di simulated annealing sono:

Congelamento:	Consiste nello stabilire il criterio di arresto dell'algoritmo.
Temperatura:	E' un valore da cui dipende la probabilità di accettare movimenti in salita. Ad ogni iterazione tale temperatura viene ridotta di una percentuale costante chiamata raffreddamento.
Numero di intorni da esplorare:	Ad ogni iterazione vengono considerati un numero (fissato) di sequenze vicine alla sequenza considerata.

Il simulated annealing è largamente applicabile nei problemi di ottimizzazione locale. Esso sembra produrre delle soluzioni migliori rispetto ad altre tecniche di ottimizzazione locale, in quanto permette di uscire da soluzioni localmente ottime ma globalmente scadenti permettendo dei movimenti in salita con una determinata probabilità.

3. UN ALGORITMO DI SIMULATED ANNEALING

Il problema da risolvere con la tecnica di simulated annealing può essere riassunto come segue: schedulare una sequenza di *n* jobs su *m* macchine stabilendo che l'ordine degli *n* jobs è lo stesso su ciascuna delle *m* macchine.

Parametri fondamentali per la fase di simulated annealing sono:

- **Inizializzazione**. Scelta di una soluzione iniziale. In teoria la scelta di una soluzione iniziale non sortisce alcun effetto sulla qualità della soluzione finale, vale a dire che la soluzione converge all'ottimo globale indipendentemente dalla soluzione iniziale [Lun86]. A tale teoria vi sono tuttavia delle eccezioni verificate sperimentalmente, in cui è stato mostrato che talvolta il processo di convergenza alla soluzione ottima è più rapido se si prende come soluzione iniziale una ottenuta mediante una buona euristica [Mas87]. Tuttavia, il tempo computazionale complessivo per una soluzione approssimata che parta da una "buona" soluzione iniziale è spesso più elevato del tempo computazionale necessario per ottenere una soluzione approssimata che parta da una soluzione iniziale qualsiasi. Il motivo di quanto detto è da ricercarsi nella particolarità del meccanismo che genera le perturbazioni.
- **Selezione di un meccanismo che generi delle piccole perturbazioni,** per passare da una configurazione ad un'altra. Lo schema di perturbazione costituisce un elemento cruciale per ottenere delle buone performance [Kir83] [Cer85]. Lo schema di perturbazione scelto consiste nel considerare due numeri generati casualmente e

scambiare di posto i jobs che nella coda di schedulazione occupano le posizioni relative ai numeri casuali scelti.

La struttura scelta per la rappresentazione del job i-esimo (con $0<i\leq n-1$) è la seguente:

- *p[j]* $\forall\ 0\leq j\leq m-1$
  E' un vettore che memorizza il processing time del job sul processore *j-esimo*. Tale dato è letto da file di input.

- *tot_p*
  E' il totale dei processing time del job, ottenuto sommando i processing time del job su *tutti* i processori

- *d*
  Valore di due date del job. Tale dato è letto da file di input e viene calcolato dal generatore nel seguente modo:
  d = tot_p + (percentuale input di tot_p)

- *c*
  E' il completed time del job; il suo valore dipende dalla particolare permutazione della coda di schedulazione. Se il job è nell'*i-esima* posizione nella coda di schedulazione, allora il completed time è dato dal valore del completed time del job che lo precede nella coda di schedulazione maggiorato del proprio processing time

- *late*

  E' il valore di late work del job. Viene calcolato nel seguente modo:

Se	$c - d \leq 0$	$late = 0$.
Se	$0 < c - d \leq tot\_p$	$late = c - d$.
Se	$c - d > tot\_p$	$late = tot\_p$.

Altre informazioni utili per la comprensione dell'algoritmo sono:

- *Actual order*

E' l'ordine di schedulazione dei jobs attualmente considerato. Si conisidera che l'ordine seguente alla lettura dei dati dal file di input sia 0,1,2,...,n-1

- *Best order*
  E' l'ordine di schedulazione dei jobs che ottiene il minimo dalla funzione obiettivo. Tale valore di minimo è memorizzato nella variabile *BOF* (*B*est *O*bject *F*unction)

Parametri scelti per la fase di simulated annealing sono:

- Criterio di " *CONGELAMENTO* "
  Si ha un " congelamento " quando l'algoritmo, per *MAXITERAZIONI* volte, non compie alcun movimento nè in discesa nè in salita.

- *Temperatura*
  La temperatura iniziale per la fase di simulated annealing è fissata nel parametro *temperatura*.

- *Raffreddamento*
  Ad ogni iterazione la temperatura viene raffreddata di un parametro *raffreddamento*.

- *L*
  Numero di intorni da visitare ad ogni iterazione.

L'algoritmo proposto è costituito da due cicli nidificati. Fondamentalmente esso sceglie due numeri (*i* e *j)* interi casuali nell'intervallo *1,...,n*; permuta, nella coda di schedulazione, il job *i* ed il job *j*; calcola il valore della funzione obiettivo. Se tale valore risulta essere inferiore del valore memorizzato in BOF, viene memorizzato l'ordine di schedulazione ed aggiornato il BOF. Altrimenti, se il valore peggiora la funzione obiettivo, viene calcolata una probabilità in base alla quale viene accettato il peggioramento della funzione obiettivo.

Il programma realizzato rispetta il seguente algoritmo:

```
Considera una soluzione iniziale S.

Considera una temperatura iniziale T>0 ed azzera il contatore per congelamento.

while "non si ha un congelamento" do
  for  w:=0 to L
    Genera due indici casuali (diversi) i e j nell'insieme {0,1,...,n-1}
    Considera la coda di scheduling  S', ottenuta permutando i jobs che sono nella posizione
actual order[i]  ed actual order[j] nella coda di schedulazione  S
    Calcola il valore della funzione obiettivo s' per la coda di schedulazione  S'
    if s' ≤  s  /* MOVIMENTO IN DISCESA   */
      /* Poni  S := S', cioè permuta fisicamente i jobs */
      s := s'
      if  BOF > s'
        BOF := s'
        best order =actual order
      endif
    else
      p := min{1,exp[-(s'-s)/temperatura] }
      Sia ε un numero casuale nell'intervallo [0,1]∩R
      if  ε  > p   /* MOVIMENTO IN SALITA */
        /* Poni  S := S' */
        s := s'
      else
        incrementa contatore per congelamento.
      endif
    endif
  endfor
temperatura = raffreddamento * temperatura
endwhile.
```

Fig. 3 - L'algoritmo

Punto di forza dell'algoritmo proposto sta proprio nel calcolare il valore della funzione obiettivo senza dover permutare fisicamente i jobs *i* e *j*.

Vediamo un esempio di tale calcolo. Supponiamo di avere la situazione rappresentata in figura 4a, e di aver scelto gli indici casuali 2 e 4. La coda risultante permutando i jobs che occupano tali indici nella coda *actual order* è illustrata nella figura 4b:

actual order	0	1	2	3	4	5	6

	job 0	job 1	job 2	job 3	job 4	job 5	job 6
mac.0	10	40	80	30	50	15	70
mac.1	8	45	87	22	52	13	75
mac.2	12	30	85	20	50	18	48

tot_p	30	115	252	72	152	46	193
d	33	126.5	277.2	79.2	167.2	50.6	212.3
c	30	145	397	469	621	667	860
late	0	18.5	119.8	72	152	46	193

best order	0	1	2	3	4	5	6

Fig. 4a

actual order	0	1	4	3	2	5	6

	job 0	job 1	job 4	job 3	job 2	job 5	job 6
mac.0	10	40	50	30	80	15	70
mac.1	8	45	52	22	87	13	75
mac.2	12	30	50	20	85	18	48

tot_p	30	115	152	72	252	46	193

d	33	126.5	167.2	79.2	277.2	50.6	212.3

c	30	145	297	369	621	667	860

late	0	18.5	129.8	72	252	46	193

best order	0	1	2	3	4	5	6

Fig. 4b

In questo caso abbiamo peggiorato il valore della funzione obiettivo; infatti la coda *best order* è immutata, ma rimane comunque evidente che:

- Per i valori di Completed Time si ha
  - per i jobs 0 ed 1 sono rimasti immutati;
  - per i jobs 2 e 3 sono stati incrementati dello stesso valore (100), tale incremento è proprio pari alla differenza
  - *diff = tot_p[4] - tot_p[2]*
  - dello schema di partenza;
  - per i jobs 4,5 e 6 sono rimasti immutati.

- Per i valori di Late Work abbiamo invece:
  - immutati per i jobs 0 ed 1 nella nuova coda di schedulazione;
  - per il job che occupa la posizione 2 nella nuova coda, calcolato il nuovo completed time, quest'ultimo è poi confrontato con il totale processing time del job 4 per calcolare il nuovo valore di late work;
  - per il job 3, calcolato il nuovo valore di completed time,viene confrontata con il valore di due date del job 3 nello schema precedente;
  - per il job in posizione 4 nella nuova coda di schedulazione, il valore di completed time è immutato, per cui è sufficiente confrontarlo con il totale processig time del job 2 nella vecchia coda di schedulazione ottenendo il nuovo valore di late work;
  - per i jobs 5 e 6 i valori di late sono immutati;

In generale, se intendiamo permutare il job i con il job j (con $i < j$) i valori di late work saranno :

- immutati per i jobs nelle posizioni *0,1,...,i-1*

- si calcola il seguente valore:
  diff = coda_jobs[j].tot_p - coda_jobs[i].tot_p

- per il job in posizione i , il nuovo valore di late work è calcolato confrontando la quantità:
  diff + coda_jobs[i].c - coda_jobs[j].d
  con il valore di due date del processo j

- per i jobs in posizione *i+1,i+2,...,j-1* il valore di late work è calcolato confrontando la quantità:
  diff + coda_jobs[k].c - coda_jobs[k].d
  con il valore di due date del processo k, dove $k \in \{i+1,i+2,...,j-1\}$

- per il jobs in posizione j , il nuovo valore di late work è calcolato confrontando la quantità:
  diff + coda_jobs[j].c - coda_jobs[i].d
  con il valore di due date del processo i

- per i jobs in posizione $j+1,j+2,...,n-1$ i valori di late work sono immutati.

Possiamo quindi calcolare il valore della funzione obiettivo senza dover calcolare le tabelle relative alla nuova coda di schedulazione, ma semplicemente elaborando delle informazioni che sono già in nostro possesso senza utilizzare quantità di memoria.

4. TEST COMPUTAZIONALI

Consideriamo il programma generatore dei numeri che formano i valori dei processing time. Per ciascun job su ogni macchina viene generato un numero casuale nell'intervallo $[0,100] \cap N$ che rappresenta il valore di processing time del job sul processore. Il seme del generatore dei numeri casuali viene inizializzato di volta in volta ad un numero diverso legato al tempo di sistema. Il tempo di due date è invece calcolato facendo la somma dei valori dei processing time del job su tutte le macchine a disposizione, aumentati di una percentuale data da input. I dati vengono scritti in append utilizzando delle apposite system call.

Dati da considerarsi comuni a tutti i test sono:

- percentuale di incremento ai valori di due date = 10.
- numero di intorni da esplorare: = 10.
- numero di esecuzioni per ogni problema = 15.

Per quanto riguarda il numero di intorni da esplorare, è stato scelto il valore sopraindicato in quanto da una verifica sperimentale è stato notato che per un numero minore di intorni si ottengono dei risultati prossimi a quello ottimo, mentre per un numero maggiore di intorni da esplorare si ottengono (nei casi da noi testati) valori che spesso coincidono con l'ottimo, ma essendo computazionalmente onerosi influiscono negativamente sul tempo di esecuzione dell'algoritmo approssimato.

5. ESPERIMENTI

Il metodo di soluzione seguito è di tipo non deterministico, e si basa sulla probabilità di ottenere una soluzione ottima in relazione alla situazione attuale della funzione

obiettivo ed al possibile miglioramento legato ad un "movimento controllato nello spazio delle soluzioni ammissibili, legato al concetto di "temperatura" dell'algoritmo, sino a pervenire al punto di "congelamento", cioé ad un ottimo accettabile.

La scelta fatta nel nostro metodo prescinde dalla soluzione iniziale, e tiene piuttosto conto del tempo totale di calcolo richiesto, essendo più importante ottenere una soluzione ammissibile "quasi" ottima in un tempo ragionevole, piuttosto che tentare di arrivare alla soluzione ottima in un tempo teoricamente illimitato.

Vengono esposti alcuni casi di test per la verifica del peso computazionale in base alle ipotesi effettuate.

Indicazioni per la comprensione dei test computazionali.

La seguente tabella riassume i test eseguiti. Le indicazioni in testa alle colonne indicano:

- **n:** numero di jobs da schedulare;
- **m:** numero di macchine utilizzate;
- **F.O.:** Risultato ottimo della funzione obiettivo;
- **F.O. = S.A.:** numero di volte in cui il risultato ottenuto con l'algoritmo approssimato di simulated annealing coincide con quello ottimo;
- **T min:** tempo minimo (in secondi) impiegato dall'algoritmo di simulated annealing per ottenere un risultato coincidente con l'ottimo;
- **T max:** tempo massimo (in secondi) impiegato dall'algoritmo di simulated annealing per ottenere un risultato coincidente con l'ottimo;
- **F.O. ≠ S.A.:** numero di volte in cui il risultato ottenuto con l'algoritmo approssimato di simulated annealing differisce da quello ottimo;
- **min:** minimo impiegato dall'algoritmo di simulated annealing per ottenere un risultato non coincidente con l'ottimo;
- **T:** tempo (in secondi) impiegato dall'algoritmo di simulated annealing per ottenere un risultato non coincidente con l'ottimo;
- **Δ %:** differenza media percentuale.

n	m	F.O.	F.O. = S.A.	T min (sec.)	T max (sec.)	F.O. ≠ S.A.	min	T (sec.)	Δ %	max	T (sec.)	Δ %
10	3	1112.200	15	0.04	0.35	0	--	--	--	--	--	--
10	4	1312.600	13	0.06	0.33	2	1338.6	0.03	1.94	1353.6	0.01	3.02
10	5	2393.210	15	0.06	0.33	0	--	--	--	--	--	--
11	3	1170.000	13	0.05	0.64	2	1195.0	0.02	2.09	1195.0	0.08	2.09
11	4	1819.800	15	0.03	0.34	0	--	--	--	--	--	--
11	5	2433.900	15	0.05	0.31	0	--	--	--	--	--	--

Fig. 5 - Test computazionali

BIBLIOGRAFIA

[Pot92] Potts C.N., Van Wassenhove L.N., 1992, "Approximation algorithms for scheduling a single machine to minimize total late work". *Operation Research Letters* 11, pp.261-266.

[Bla91] Blazewicz J., Dror M., Weglarz J., 1991, "Mathematical programming formulation for machine scheduling: A survey". *European Journal of Operational Research* 51, pp. 283-300.

[Ogb90] Ogbu F.A., Smith D.K., 1990, "The application of the simulated annealing algorithm to the solution of the $n/m/C_{\max}$ flowshop problem". *Computer & Operations Research* Vol 17, No. 3, pp 243-253.

[Bla87] Blazewicz J 1987, "Selected topics in scheduling theory". *Annals of Discrete Mathematics* 31, pp. 1-60.

[Lun86] M. Lundy and A. Mess, "Convergence of an annealing algorithm". Prog.34, 111-124 (1986).

[Hey86] I. Heynderickx, H de Raedt and Schoemaker, "Simulated annealing method for the determination of spin Hemiltonian parameters from electron-spin-resonance data". *J. Magnet. Resonance 70*, 134-139 (1986).

[Aar85] E.H.L. Aarts and P.J.M. Van Laarhoven, "Statistical cooling: a general approach to combinatorial optimisation problems". *Philips J Res. 40*, 193-226 (1985).

[Cer85] V. Cerny, "A termodynamical approach to the travelling salesman problem: an efficient simulation algorith". *J. Optimisat. Teory Applic.* 45, 41-51 (1985).

[Lun85] M. Lundy, "Applications of the simulated annealing algorithm to combinatorial problems in statistics". *Biometricka 72*, 191-198 (1985).

[Kir83] S. Kirkpatrick, C.D. Gelatt Jr and M.P. Vecchi, “Optimization by simulated annealing”. *Science 220*, 671-680 (1983)

[Vec83] M.P. Vecchi and S. Kirkpatrick, “Global wiring by simulated annealing”. *IEEE Trans. Computer-Aided Design CAD-2*, 215-22 (1983).

[Gar79] Garay M.R., Johnson D.S. “Machine Scheduling Problems: Classification, Complexity and Computation”. *Martinus Nijhoff, The Hague*, 1976.

[Rin76] Rinnooy Kan A.H.G, "Machine Scheduling Problems: Classification, Complexity and Computations". *Martinus Nijhoff*, *The Hague,* 1976.

[Hor74] Horn W.A., 1974, "Some simple scheduling algorithm". *Naval Research Logistics Quarterly* 21, pp. 177-185.

[Met53] W. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, “Equation of state calculations by fast computing machines”. *J. Chem. Phys* 21, 1087-1092 (1953)