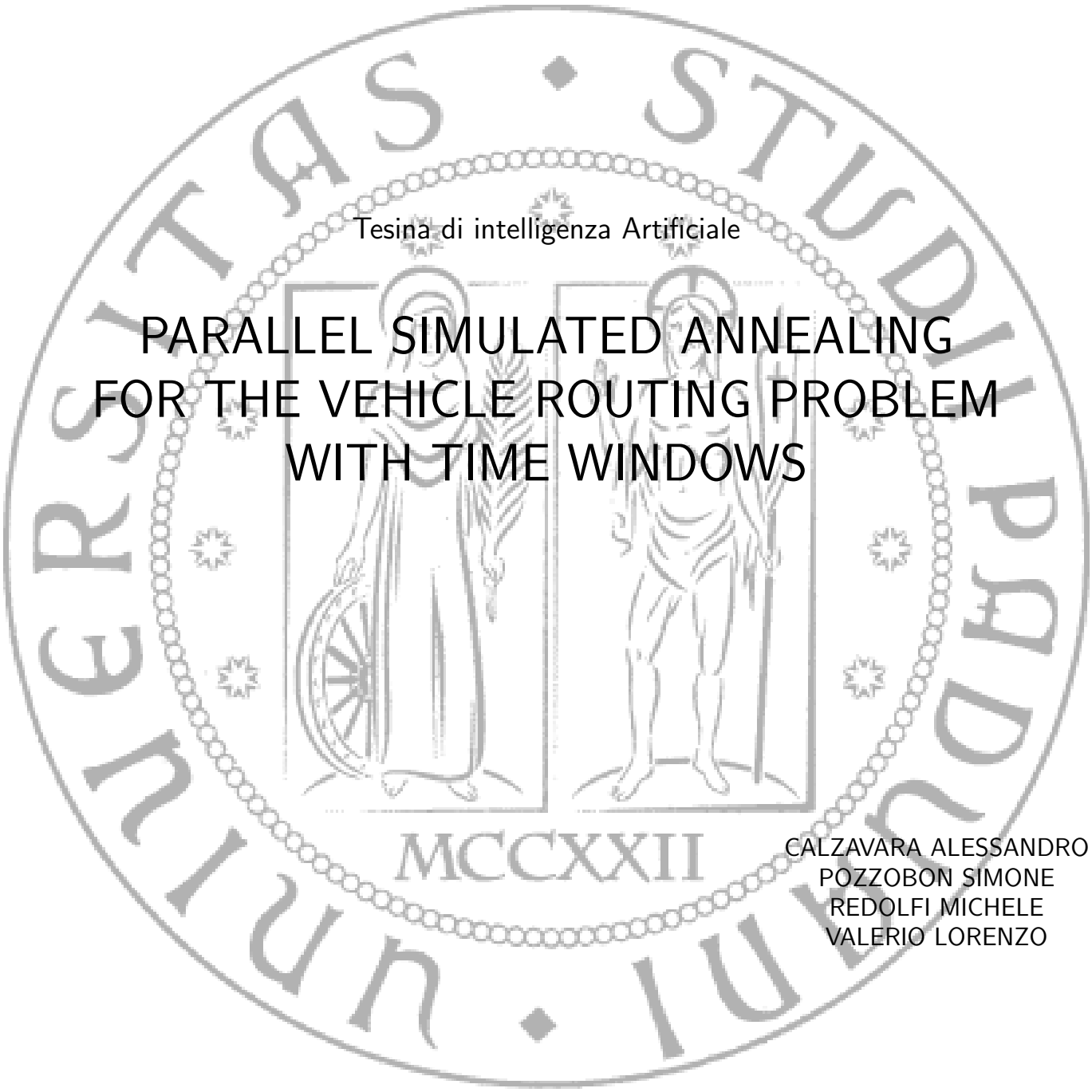
UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica



Tesina di intelligenza Artificiale

PARALLEL SIMULATED ANNEALING FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

CALZAVARA ALESSANDRO
POZZOBON SIMONE
REDOLFI MICHELE
VALERIO LORENZO

ANNO ACCADEMICO 2010-2011

Parallel Simulation Annealing for a Vehicle Routing Problem with Time Windows

Abstract

In questa tesina é presentato un modo per risolvere il Vehicle Routing Problem with Time Windows usando una implementazione parallela dell'algoritmo di Simulated Annealing. Il nostro obiettivo é quello di cercare le migliori soluzioni, in termini di minimizzazione del numero di veicoli e, in secondo luogo, della distanza percorsa da ognuno di essi, su istanze giá note in letteratura usando appunto il parallelismo.

1 Introduzione

Questa sezione é dedicata alla definizione del Vehicle Routing Problem e ad una breve spiegazione dell'approccio di Simulated Annealing.

1.1 Vehicle Routing Problem (VRP)

Il **Vehicle Routing Problem** (VRP) é un tipico problema operativo nelle reti di distribuzione e consiste nello stabilire i percorsi di una serie di veicoli per servire un insieme di clienti. Quindi si tratta di, dato un insieme di veicoli con determinate caratteristiche di carico, servire un insieme di clienti (anche essi con determinate caratteristiche) distribuiti all'interno di una rete di trasporto a partire da uno (o piú) depositi centrali. Possibili applicazioni del VRP sono frequenti in problemi logistici e distributivi di dettaglio, nella raccolta di rifiuti industriali,nei percorsi dei mezzi pubblici, nei servizi di ronda, nei trasporti bancari e postali, ecc.. . Per esempio, si puó pensare a contesti applicativi in cui si debba distribuire al dettaglio (oppure provvedere alla raccolta) un certo bene. In applicazioni reali sono presenti frequentemente molti vincoli aggiuntivi che possono complicare la struttura del problema.

A seconda di come sono configurati i clienti, si possono distinguere due classi:

1. problemi di *arc-routing* quando i clienti sono uniformemente distribuiti lungo le linee di connessione;

2. problemi di *node-routing* quando i clienti possono essere rappresentati come entitá distinte, ad esempio i nodi di un grafo.

La classe di problemi piú presente nel mondo reale é quella di *node-routing*. Possiamo fare una prima distinzione in base alla disponibilitá di carico di un veicolo. Se tale disponibilitá supera la richiesta, allora il problema si riduce a quello del *Commesso Viaggiatore (TSP)*. Altrimenti, nel caso in cui la somma delle richieste di tutti i clienti risulti superiore alla capacitá di trasporto di un singolo veicolo si parla allora propriamente di problemi VRP.

Nella sezione successiva vedremo in dettaglio le specifiche di problema che abbiamo adottato.

1.2 Simulated Annealing

Il **Simulated Annealing** (SA) é divenuto negli ultimi anni una metaeuristica ampiamente discussa. L'idea originale nasce negli anni 50 ma diventa effettivamente operativa nei problemi di ottimizzazione negli anni 80. Si é provato che questa strategia converge ad una soluzione ottima con una probabilitá vicina ad uno,

ma a costo di una velocitá di convergenza che puó essere anche esponenziale (anche se il suo comportamento asintotico puó essere approssimato in un tempo polinomiale).

Il processo di annealing é tipico della statistica meccanica ed é un processo usato per portare ad una configurazione cristallina ordinata un metallo che passa dallo stato liquido allo stato solido. Questo processo consiste nell'abbassare gradualmente e in maniera controllata la temperatura di raffreddamento del metallo. Ad alte temperature gli atomi del sistema si trovano in uno stato altamente disordinato e quindi l'energia del sistema é molto alta. Un'abbassamento della temperatura troppo drastico puó causare difettositá del reticolo cristallino provocando rotture del metallo stesso.

La metaeuristica SA implementa questa tecnica garantendo una minore probabilitá di bloccarsi in un ottimo locale. In questo caso la temperatura viene vista come un parametro per decidere la probabilitá di accettazione di una soluzione trovata.

In generale, inizialmente viene fatta una mossa casuale. Se la mossa successiva migliora la situazione allora viene sempre accettata, altrimenti viene scelta con qualche probabilitá inferiore a 1. Tale probabilitá decresce con la temperatura T che scende costantemente: ció permette di accettare mosse cattive piú facilmente all'inizio in condizione di alta temperatura (comportandosi piú o meno come una *random search*); tali scelte diventeranno via via meno probabili con la diminuzione della temperatura, e si localizzeranno in una zona del dominio maggiormente promettente.

Nella prossima sezione saremo piú chiari nelle scelte dei parametri del SA che abbiamo adottato.

2 Formulazione del problema

In questa sezione, definiamo le specifiche e i parametri del problema trattato.

Nella versione base del problema VRP per il trasporto di merce, ad ogni cliente viene solamente associato il quantitativo di merce che deve essere ritirata o consegnata. Nel nostro caso aggiungiamo anche un'altra specifica: ad ogni cliente é associata una *time window*, cioé ogni veicolo che deve portare a termine il suo compito deve arrivare entro una certa finestra temporale. Se il veicolo arriva prima dell'inizio della finestra temporale, dovrá aspettare, mentre se arriva in ritardo andrá via senza portare a termine l'operazione. Si puó pensare ad esempio ai negozi che hanno come orario di consegna della merce le ore di apertura del negozio stesso. Un'altra dimensione per la classificazione di un VRP é data dalle caratteristiche dei veicoli. Nel nostro esempio usiamo una *flotta omogenea*, in cui i veicoli che la compongono sono tutti identici per quantitá di carico.

La rete di trasporto utilizzata é di tipo *euclideo*, cioé le distanze tra i nodi rispettano la diseguaglianza triangolare $d_{ij} \leq d_{ik} + d_{kj}$. Ogni cliente é caratterizzato da due coordinate che ne definiscono la sua posizione nel piano cartesiano permettendo cosí di calcolare le distanze tra clienti come le distanze dei punti nel piano (si parla in questo caso di istanze planari). Inoltre, supporremo che i grafi rappresentanti le infrastrutture di trasporto siano grafi completi. Nel caso in cui un arco di collegamento tra due nodi non dovesse essere presente supporremo la presenza di un costo infinito associato all'arco.

Per descrivere il problema, assumiamo la presenza di n clienti che devono essere serviti. Allora abbiamo per ogni cliente i:

q_i : quantitá di carico richiesta da ogni cliente;

s_i : tempo di servizio richiesto;

e_i : limite minimo di tempo in cui il servizio puó iniziare presso il cliente;

f_i : limite massimo di tempo in cui il servizio puó iniziare presso il cliente;

t_{ij} : tempo di viaggio tra 2 clienti;

d_{ij} : distanze minime tra i nodi ($i = 1$ é il deposito);

b_i : tempo in cui il servizio puó iniziare presso il cliente;

b_{ij} : tempo corrente quando il servizio puó iniziare su j, dato che il cliente j é inserito dopo i nel percorso;

w_{ij} : tempo d'attesa

Q : capacitá di ogni veicolo.

Riassumendo, il nostro problema consiste nel trovare un insieme di strade che iniziano e terminano presso un deposito che serve un insieme di consumatori. C'é un insieme di veicoli, tutti con la stessa capacitá di trasporto, ed ogni mezzo non puó servire piú clienti di quanto gli permetta la sua capacitá. Ad ogni cliente é associata una finestra temporale ed un tempo di servizio. Un cliente i é servito da un singolo veicolo esattamente una volta, entro la finestra temporale $[e_i, f_i]$. b_{ij}, il tempo in cui il servizio puó iniziare presso il cliente j, dato che il cliente j é inserito immediatamente dopo il cliente i nel percorso, viene calcolato come:

$$b_{ij} = max\{e_j, b_i + s_i + t_{ij}\} \tag{1}$$

Un tempo d'attesa:

$$w_{ij} = e_j - (b_i + s_i + t_{ij}) \tag{2}$$

é richiesto se un veicolo arriva presso il cliente j prima di e_j. L'obiettivo é dunque quello di stabilire un insieme di strade che coprono ogni cliente esattamente una volta, assicurando che il servizio per ogni cliente inizia dentro la finestra temporale e preserva i vincoli capacitivi del mezzo di trasporto. Inoltre l'insieme delle strade deve minimizzare il numero di veicoli usati e la distanza totale coperta dai veicoli.

L'applicazione del SA per la risoluzione del VRPTW é la seguente. Inizialmente una soluzione al problema é presa come migliore soluzione. Ad ogni passo una soluzione prossima é determinata o muovendo il cliente migliore (in termini di costo) da un percorso in una migliore posizione in un altro percorso oppure selezionando il miglior cliente e spostandolo in una posizione migliore all'interno dello stesso percorso.

É di primaria importanza per l'effettivitá e l'efficienza delle euristiche per questo problema il modo in cui i vincoli sulla finestra temporale sono incorporati nel processo di soluzione. Come prima cosa infatti esaminiamo le condizioni necessarie e sufficienti per la fattibilitá temporale dopo l'inserimento di un cliente u tra 2 clienti i_{p-1} e i_p[1], con $1 \leq p \leq m$, in un percorso $(i_0, i_1, i_2, ..., i_m = i_0)$. Assumiamo innanzitutto che ogni veicolo lasci il deposito il prima possibile. Dopo aver creato l'intera schedulazione dei veicoli, possiamo aggiustare i tempi di partenza separatamente per ogni veicolo in modo da eliminare i tempi di attesa non necessari.

Sia $b_{i_p}^{new}$ il nuovo istante nel quale il servizio presso i_p inizia dato l'inserimento di u e sia w_{i_r} il tempo d'attesa su i_r con $p \leq r \leq m$. L'inserimento allora causa un tempo di *push forward* nella schedulazione di i_r:

$$PF_{i_p} = b_{i_p}^{new} - b_{i_p} \geq 0 \tag{3}$$

$$PF_{i_{r+1}} = max\{0, PF_{i_r} - w_{i_{r+1}}\}, \qquad p \leq r \leq m-1 \tag{4}$$

Se $PF_{i_p} > 0$ allora alcuni clienti i_r, $p \leq r \leq m$ possono non essere piú schedulabili. Esaminiamo questi clienti sequenzialmente per la schedulabilitá finché:

- troviamo un i_r per cui $PF_{i_r} = 0$, oppure
- i_r non é schedulabile, oppure
- tutti i clienti i_r , $p \leq r \leq m$ sono stati esaminati

Inoltre perché sia soddisfatta (3) é importante la validitá della disuguaglianza triangolare.

Abbiamo cosí dimostrato che le condizioni necessarie e sufficienti per la schedulabilitá dopo l'inserimento di un cliente u in un percorso schedulabile $(i_0, i_1, .., i_m)$ sono $b_u \leq f_u$ e $b_{i_r} + PF_{i_r} \leq f_{i_r}$, con $p \leq r \leq m$

L'euristica scelta per la generazione di una soluzione iniziale feasible é l'euristica di inserimento di Solomon. Essa genera sequenzialmente i percorsi inserendo i clienti in modo greedy usando tre regole di inserimento per mantenere la schedulabilitá durante il processo di inserzione:

1. selezione del prossimo cliente basata sul piú piccolo e_j;
2. regola basata sulla strettezza della time window calcolata come: $100 \cdot (f_j - e_j) - d_{0j}$ con d_{0j} distanza di j dal deposito
3. regola basata sul piú grande valore di d_{ij}

[1] usiamo questa terminologia per semplificare l'illustrazione del problema

La locazione di inserzione migliore per un dato cliente é determinata dall'incremento del tempo di viaggio e della lunghezza del percorso. Consideriamo di inserire un cliente j tra i clienti i e k in un percorso r. Usiamo allora queste misure di costo :

$$c_1(i,j,k) = d_{ij} + d_{jk} - d_{ik} \tag{5}$$

$$c_2(i,j,k) = b_{jk} - b_k; \tag{6}$$

Dove (5) calcola l'incremento della distanza e (6) l'incremento locale del tempo di schedulazione. Introduciamo il criterio finale:

$$c_3(r) = \alpha_1 c_1(i,j,k) + \alpha_2 c_2(i,j,k), \qquad \alpha_1 + \alpha_2 = 1 \tag{7}$$

Il cliente j é inserito nel percorso r con $r = argmin\{c_3(v); v = 1,2,...,N\}$ con N numero dei veicoli. Usando i parametri ($\alpha_1 = 1, \alpha_2 = 0$), ($\alpha_1 = 0, \alpha_2 = 1$) piú le 3 regole prima specificate, vengono generati 6 potenziali inserimenti in ogni run per determinare la soluzione migliore. Tali inserimenti vengono ordinati in base all'incremento di costo sulla soluzione crescente. Se nessuno di questi inserimenti risulta schedulabile rispetto alla capacitá del veicolo e della finestra temporale, allora viene generata un nuova rotta finché ci sono dei clienti ancora da schedulare.

Dopo aver generato la soluzione iniziale, usando il simulated annealing in parallelo [2] la soluzione iniziale viene raffinata. Ogni processo accetta sempre una soluzione che ha un costo inferiore rispetto alla soluzione precedente, altrimenti viene accettata con probabilitá pari a :

$$T_i/(T_i + \delta) \tag{8}$$

dove:

- T_i é la temperatura di raffreddamento, con $T_0 = \gamma \cdot cost(s_0)$ (dove γ é una costante e s_0 costo della situazione iniziale) e che diminuisce secondo la regola $T_{i+1} = \beta T_i$;
- δ é l'incremento del costo della soluzione.

L'equazione mostra come una soluzione con un incremento del costo é piú facilmente accettata se la temperatura é alta (*uphill moves*), e quindi, essendo la temperatura decrescente, molte *uphill moves* saranno scartate. Il fatto di poter prendere in considerazione anche soluzioni a costo maggiore permette di evitare, come detto prima, l'entrata prematura in un regione d'ottimo locale.

Il valore del costo della soluzione é dato da:

$$cost(s) = d + \sigma(cn + e_{min}) \tag{9}$$

dove:

- d é la distanza totale di viaggio in un percorso ;
- σ é una costante, pari ad un valore che permetta $\sigma(cn + e_{min}) \gg d$ (si vuole infatti ottimizzare il numero di percorsi);
- c é il numero di percorsi (=numero di veicoli);
- e_{min} é il numero di clienti nel percorso piú corto

L'algoritmo sequenziale termina quando l'equilibrio é raggiunto, cioé quando τ riduzioni di temperatura non portano a nessun miglioramento. La soluzione presa non é l'ultima che viene ottenuta, ma la migliore durante tutto il processo.

La complessitá temporale dell'algoritmo sequenziale é pari a

$$T(n) \leq an^3 = O(n^3) \tag{10}$$

L'uso di SA in parallelo serve per ottenere una migliore accuratezza (intesa come prossimitá alla soluzione globale) della soluzione del problema. I p processi $P_1, P_2, ..., P_p$ collaborano ogni ω passi passandosi la migliore soluzione trovata. Siano:

[2] rimandiamo le specifiche al paragrafo dedicato al pseudocodice dell'algoritmo

- $V_r^{(j)}(T)$ con $j = 1, 2, ..., p$ e $r = 1, 2, ..., r_{max}$, la catena di Markov di ogni processo;
- $P_T^{(V)}$ la realizzazione di un passo della catena alla temperatura T e allo stato di partenza P;
- $\overline{V}_r^{(j)}$ la migliore soluzione trovata dai processi j tra il passo 1 e r

Assumiamo il seguente schema di cooperazione:

$$V_{r+1}^{(1)} = P_T(V_r^{(1)}) \tag{11}$$

$$V_{r+1}^{(j)} = P_T(V_r^{(j)}) \tag{12}$$

$$V_{u\omega}^{(j)} = \begin{cases} P_T(V_{u\omega-1}^{(j)}), & \text{se } cost(P_T(V_{u\omega-1}^{(j)})) \leq cost(\overline{V}_{u\omega}^{(j-1)}) \\ \overline{V}_{u\omega}^{(j-1)}, & \text{altrimenti} \end{cases} \tag{13}$$

La migliore soluzione trovata da l-esimo processo verrá propagata in avanti. I processi interagiscono frequentemente con il valore di temperatura costante.

Algoritmo

Possiamo suddividere l'algoritmo Parallel SA in due parti: una prima parte chge riguarda il processo coordinatore P_0 seguita da una seconda parte riguarda i processi P_j, $j = 1, ..., p$. Qui sotto vediamo il pseudocodice per il processo principale P_0:

```
1:  Prendi la initial_solution come migliore soluzione conosciuta del problema
2:  Invia la initial_solution ai processi P_j, j = 1, 2, ..., p
3:  final_solution := initial_solution
4:  equilibrium_counter := 0
5:  while equilibrium_counter ≤ τ do
6:     Ricevi best_local_solution_j dai processi P_j, j = 1, 2, ..., p
7:     Scegli best_global_solution tra i best_local_solution_j con , j = 1, 2, ..., p
8:     if cost(best_global_solution) < cost(final_solution) then
9:        final_solution := best_global_solution
10:       equilibrium_counter := 0
11:    else
12:       equilibrium_counter := equilibrium_counter + 1
13:    end if
14:    go := (equilibrium_counter ≤ τ)
15:    invia go ai processi P_j, j = 1, 2, ..., p
16: end while
17: return final_solution
```

Il processo P_0 si occupa di scegliere la miglior soluzione globale tra le soluzioni locali che riceve dagli altri processi (linea 7). Tale soluzione viene testata con la soluzione precedentemente memorizzata; se é migliore, viene memorizzata, altrimenti viene aumentato il contatore che determina la *stopping condition* della ricerca (linee 8 - 13).

Qui sotto il pseudocodice per i processi P_j, $j = 1, ..., p$:

```
1: Ricevi initial_solution da P_0
2: old_solution_j := initial_solution
3: best_local_solution_j := initial_solution
4: T := γ · cost(initial_solution) {temperatura iniziale}
5: loop
6:    for iteration_counter_j := 1 to n^2 do
7:       annealing_step(old_solution_j, best_solution_j)
```

8: **if** numero di passi di annealing sono un multiplo di n **then**
9: **if** $j = 1$ **then**
10: invia $best\_local\_solution_1$ al processo P_2
11: **else**
12: ricevi $best\_local\_solution_{j-1}$ dal processo P_{j-1}
13: **if** $cost(best\_local\_solution_{j-1}) < cost(best\_local\_solution_j)$ **then**
14: $best\_local\_solution_j = best\_local\_solution_{j-1}$
15: **end if**
16: **if** $j < p$ **then**
17: invia $best\_local\_solution_j$ al processo P_{j+1}
18: **end if**
19: **end if**
20: **end if**
21: **end for**
22: Invia $best\_local\_solution_j$ a P_0
23: Ricevi go da P_0
24: **if** $go = false$ **then**
25: stop
26: **end if**
27: $T := \beta \cdot T$
28: **end loop**

I processi P_j, $j = 1, ..., p$ si occupano di compiere la ricerca di annealing usando la stessa soluzione iniziale e schema di raffreddamento usato dall'algoritmo sequenziale. Ad ogni temperatura il processo P_j esegue n^2 passi di annealing (linea 7). I processi cooperano ogni $\omega = n$ passi, passandosi le loro soluzioni migliori (linea 8 - 20). Al completamento degli n^2 passi di annealing i processi inviano le loro soluzioni migliori al processo P_0 (linea 22) che si occuperá di trovare la soluzione migliore.

Si nota che l'equilibrio é raggiunto se $\tau n^2 p$ passi di annealing eseguiti dai processi durante τ diminuzioni di temperatura consecutivi non cambiano la soluzione finale.

Rispetto all'algoritmo sequenziale, quello parallelo ha un fattore di complessitá maggiore rispetto a quello sequenziale, a causa delle comunicazioni tra i processi $P_1, P_2, ..., P_p$ e tra P_0 e $P_1, P_2, ..., P_p$.

Consideriamo ad esempio l'ultimo processo P_p: il numero di comunicazioni all'interno del ciclo for sono pari ad n mentre il costo di una singola comunicazione é proporzionale a $(p-1)n$ in quanto i messaggi sono di dimensione $O(n)$. Quindi il costo dell'esecuzione del ciclo for nel processo P_p é pari a $n^3 + (p-1)n^2$ e, considerando anche il processo P_0, il costo totale dell'esecuzione non é superiore a $pn + a(n^3 + (p-1)n^2)$, dove a é il numero di cicli while compiuti da P_0 e pn é il costo delle comunicazioni iniziali (linea 1) e finali (linee 22 - 23) tra P_0 e i P_j.

Allora, nel caso peggiore, la complessitá del Simulated Annealing parallelo per VRPTW é:

$$\mathcal{T}_p(n) \leq O(n^3 + pn^2) \tag{14}$$