UNIVERSITÀ DEGLI STUDI DI PADOVA
Facoltà di Ingegneria
Corso di laurea in Ingegneria Informatica

Tesi di Laurea

Algoritmi euristici per il Problema del Commesso Viaggiatore con Finestre Temporali

Candidato:
Nicola Menon

Relatore:
Prof. Matteo Fischetti

Co-relatore:
Prof. Michele Monaci

Anno accademico 2013-2014

Introduzione

Il Problema del Commesso Viaggiatore con Finestre Temporali (in inglese Traveling Salesman Problem with Time Windows, TSPTW) è una variante del più noto problema del Commesso Viaggiatore (TSP) . Dato un grafo G formato da n nodi ed altrettante finestre temporali associate ai nodi, il TSPTW può essere definito come il problema di trovare un circuito hamiltoniano che tocca una sola volta tutti gli n clienti, che inizia e termina in un nodo di partenza e che soddisfa tutti i vincoli delle finestre temporali minimizzando il costo totale dato dalla somma dei costi degli archi percorsi.

TSPTW è un problema NP-hard in quanto si tratta di una generalizzazione del problema del Commesso Viaggiatore, che è anch'esso NP-hard. Quindi risulta utile un euristico in grado di risolvere in modo efficace le istanze realistiche più grandi in tempi ragionevoli. In questo lavoro si descrive l'applicazione di due algoritmi euristici: uno basato su un modello matematico chiamato Proximity Search, l'altro rientra nella categoria dei meta-euristici ed è il Simulated Annealing. Entrambi sono dei metodi di risoluzione generali che fanno poche ipotesi sul problema preso in esame. Nel nostro caso li utilizziamo per la risoluzione del problema del TSPTW, unendo i due euristici in un algoritmo che chiamiamo SAPS (Simulated Annealing - Proximity Search) e confrontando i risultati ottenuti con quelli di altri algoritmi euristici specifici per il suddetto problema, quali GVNS (General Variable Neighbourood Search) o CAH (Compressed Annealing Heuristic). Infine valuteremo l'efficacia del Proximity Search come procedura di raffinamento generale per soluzioni ottenute da algoritmi euristici.

Indice

Elenco delle tabelle

Capitolo 1

TSPTW

In questo capitolo viene descritto il problema TSPTW, il suo modello matematico ed il preprocessing applicato al problema

1.1 Descrizione TSPTW

L'idea di base del problema del Commesso Viaggiatore con Finestre Temporali (TSPTW) è quella di trovare, dato un insieme di clienti, un tour che parte e finisce in un determinato nodo di partenza e che visita ciascun cliente all'interno della sua finestra temporale. In letteratura esistono molte varianti di questo problema, in questa tesi si fa riferimento alla definizione data da Da Silva - Urrutia [DaS10].

Il problema TSP con Finestre Temporali consiste nel trovare un tour di costo minimo con un nodo di inizio e di fine fissato che visita tutti i clienti. Ad ogni cliente è associato:

- un tempo di servizio β_i: questo è il tempo necessario a servire il cliente
- un tempo di apertura della finestra temporale a_i: istante prima del quale non è possibile servire il cliente
- un tempo di chiusura della finestra temporale b_i: il servizio al cliente non può iniziare dopo questo istante

Ogni nodo va visitato una ed una sola volta. Il costo degli archi che uniscono i nodi rappresenta il tempo di percorrenza e nel nostro caso anche il tempo di servizio. Ad esempio il costo dell'arco tra due nodi i e j sarà la somma del tempo di servizio di i e del tempo di percorrenza da i a j. Il costo totale di un tour è la somma dei costi degli archi utilizzati nel tour. Il tempo di apertura e di chiusura di un nodo i definiscono la finestra temporale $[a_i, b_i]$ all'interno della quale deve iniziare il servizio al cliente. E' possibile visitare un cliente prima dell'apertura della sua finestra temporale ma per il servizio bisogna aspettare fino alla sua apertura. I tempi di chiusura invece devono essere rispettati, quindi un tour che non raggiunge ciascun cliente prima della chiusura della rispettiva finestra temporale viene considerato non ammissibile (infeasible).

Alcuni autori (ad esempio Dash, Günlük, Lodi e Tramontani [Dash10]) ad ogni arco assegnano due valori: un tempo di percorrenza ed un costo. Il costo viene usato al posto del tempo di percorrenza nella funzione obiettivo. Tuttavia nella maggioranza dei problemi usati in letteratura ed in tutti quelli considerati in questa tesi viene utilizzato solamente il tempo di percorrenza. In ogni caso il tempo totale di percorrenza del tour generalmente è diverso dal valore della funzione obiettivo, in quanto quest'ultima non tiene conto dei tempi di attesa derivanti dal possibile arrivo in un nodo in un istante antecedente l'apertura della finestra temporale.

La difficoltà pratica del TSPTW sta nel fatto che, al contrario del TSP semplice dove sono state risolte istanze da più di 80000 nodi, solo istanze di circa 100 nodi sono state risolte all'ottimo, mentre gli euristici affrontano istanze con non più di 400 nodi. La complessità del problema non dipende solo dal numero di nodi ma anche dalla qualità delle finestre temporali, cioè dalla loro ampiezza media rapportata al tempo di percorrenza medio. Questa caratteristica emerge chiaramente nei test effettuati con le classsi di problemi Dumas, Gendreau, Da Silva - Urrutia e Langevin (dal nome degli autori che le propongono per la prima volta), nelle quali ci sono istanze che differiscono fra loro solamente per l'ampiezza media delle finestre temporali dei loro nodi.

1.2 Modello Matematico

Viene dato un grafo completo $G = (V, A)$, dove $V = \{0, 1, 2, ..., n\}$ è l'insieme dei nodi da raggiungere. Il nodo 0 é il nodo di partenza, mentre $A = \{(i, j) : i, j \in V\}$ è l'insieme degli archi del grafo. Ogni nodo i ha una finestra di tempo associato $[a_i, b_i]$ dove a_i e b_i rappresentano rispettivamente il tempo di apertura e chiusura della finestra temporale. Ad ogni arco (i, j) è associato un costo c_{ij} e un tempo di percorrenza d_{ij}. In tutti i problemi considerati in questa tesi per ogni $i, j \in V$ si ha che $c_{ij} = d_{ij}$. Il problema sarà composto dalle variabili binarie x_{ij} associate ai lati e dalle variabili intere t_i che rappresentano l'istante di ingresso nel nodo i. Nel nostro modello non sono presenti i tempi di servizio; questi come spiegato nella sezione precedente si possono considerare come compresi nei tempi di percorrenza. Il modello matematico dal quale partiamo è questo:

$$
\begin{cases}
\min \sum_{i,j \in A} c_{ij} x_{ij} & & (1) \\
\sum_{i \in \delta^-(h)} x_{ij} = \sum_{i \in \delta^+(h)} x_{ij} = 1 & \forall h \in V & (2) \\
t_j \geq t_i + d_{ij} - M(1 - x_{ij}) & \forall (i, j) \in A & (3) \\
x_{ij} \in \{0, 1\} & \forall (i, j) \in A & (4) \\
a_i \leq t_i \leq b_i & \forall i \in V & (5) \\
t_0 = 0 & & (6) \\
t_i \geq 0 & \forall i \in V / \{0\} & (7)
\end{cases}
$$

La funzione obiettivo (1) minimizza il costo del tour. I vincoli (2) rappresentano il fatto che ogni nodo deve essere visitato una ed una sola volta. I vincoli (3) assicurano che il tempo di arrivo ad un nodo j non sia inferiore al tempo di arrivo nel nodo immediatamente precedente più il tempo di percorrenza fra i due nodi. Dato che per (6) e (7) i tempi di arrivo sono positivi per tutti i nodi diversi dal nodo di partenza, i vincoli (3) evitano la formazione dei sottocicli, quindi a differenza del TSP semplice non abbiamo bisogno dei vincoli di subtour elimination (SEC). I vincoli (4) definiscono le variabili x_{ij} come binarie, infine i (5) assicurano che i tempi di entrata nei nodi siano compatibili con le finestre temporali.
I vincoli (3) possono essere scritti in alternativa alla formulazione big M come indicator constraints, cioè come dei vincoli attivi solo se una variabile assume un

certo valore. In questo caso i vincoli (3) diventano

$$x_{ij} = 1 \rightarrow t_j \geq t_i + d_{ij} \quad \forall (i,j) \in A \quad (3)$$

1.3 Preprocessing

Prima di risolvere il problema andiamo a modificare il grafo $G = (V, A)$ eliminando alcuni archi e ridimensionando alcune finestre temporali, ottenendo un'istanza equivalente. Il procedimento seguente viene descritto da Ascheuer [Asch01].

1.3.1 Ridimensionamento delle finestre temporali

Procediamo con il ridimensionamento delle finestre temporali iterando i seguenti 4 step finchè non otteniamo più cambiamenti nelle finestre temporali. Ricordiamo che a_i e b_i rappresentano gli estremi della finestra temporale del nodo i, d_{ij} il tempo di percorrenza dell'arco (i,j) mentre $V^+(i)$ e $V^-(i)$ rappresentano gli insiemi dei nodi connessi ad i da archi rispettivamente uscenti ed entranti nel nodo i.

$$Step1.\ a_k \rightarrow max\left\{a_k, min_{i \in V^-(k)}\left\{a_i + d_{ik}\right\}\right\} \qquad \forall k \in V : V_-(k) \neq \emptyset$$
$$Step2.\ a_k \rightarrow max\left\{a_k, min\left\{b_k, min_{i \in V^+(k)}\left\{a_i + d_{ki}\right\}\right\}\right\} \qquad \forall k \in V : V^+(k) \neq \emptyset$$
$$Step3.\ b_k \rightarrow min\left\{b_k, max\left\{a_k, max_{i \in V^-(k)}\left\{b_i + d_{ik}\right\}\right\}\right\} \qquad \forall k \in V : V^-(k) \neq \emptyset$$
$$Step4.\ b_k \rightarrow min\left\{b_k, max_{i \in V^+(k)}\left\{b_i + d_{ki}\right\}\right\} \qquad \forall k \in V : V_+(k) \neq \emptyset$$

1.3.2 Costruzione della lista di precedenze dei nodi

Dopo aver ridimensionamento le finestre temporali costruiamo una lista con le relazioni di precedenza fra i nodi. Se un nodo $i \in V$ deve essere visitato prima di $j \in V$ in ogni tour ammissibile, allora scriviamo $i \prec j$. Per prima cosa definiamo $0 \prec j$ per tutti i nodi $j \in V \setminus \{0\}$. Poi definiamo $i \prec j$ per tutte le coppie $i, j \in V \setminus \{0\}$ tali che $a_j + d_{ji} > b_i$. Quindi ripetiamo lo step

$$for\ i,j,k \in V \setminus \{0\}, if\ i \prec j \wedge j \prec k \rightarrow i \prec k$$

finchè non troviamo altre relazioni di precedenza.

1.3.3 Eliminazione archi

Dopo aver ridimensionato le finestre temporali ed aver costruito la lista delle precedenze, procediamo con l'eliminazione di archi in A usando implicazioni logiche. Innanzitutto eliminiamo gli archi $(i, j) \in A$ tali che $j \prec i$ e tutti gli $(i, j) \in A$ tali che $i \prec k \prec j$ per un $k \in V \setminus \{0\}$. Quindi per tutti gli archi $(i, j) \in A$ rimanenti e per ogni $k \in V \setminus \{0\}$ verifichiamo se la seguente condizione sia vera:

$$\begin{aligned}(k \prec i \ \vee \ k \prec j \ \vee \ a_i + d_{ij} + d_{jk} > b_k) \wedge \\ (i \prec k \ \vee \ j \prec k \ \vee \ a_k + d_{ki} + d_{ij} > b_j)\end{aligned}$$

Se questa condizione viene verificata, si può concludere che (i, j) non può far parte di un tour ammissibile, quindi lo eliminiamo da A.

Capitolo 2

Proximity Search

In questo capitolo descriviamo l'algoritmo euristico chiamato Proximity Search [Fis13], che può essere utilizzato per tutti i problemi di programmazione intera mista. La programmazione intera mista (Mixed Integer Programming – MIP) è uno strumento per modellare e risolvere problemi di ottimizzazione complessi. Poichè la risoluzione di problemi MIP rientra nella classe dei problemi NP - hard, l'introduzione di metodi di tipo euristico per affrontare questi problemi diventa molto importante. Nel nostro caso il problema NP - hard considerato sarà il TSP con Finestre Temporali.

2.1 Descrizione algoritmo

Un generico modello MIP si presenta nella forma :

$$\begin{cases} min\ c^T x \\ Ax \geq b \\ x \geq 0 \\ x_j\ intera\ \forall j \in I \end{cases}$$

dove $c^T x$ rappresenta la funzione obiettivo del problema, $Ax \geq b$ sono un insieme di vincoli lineari che devono essere soddisfatti nella soluzione del problema, e le variabili $x \in \mathbb{R}_n$ sono non negative mentre le componenti x_j devono essere intere per gli $j \in I \subseteq \{1, ..., n\}$. Gli euristici MIP-based sono algoritmi che sfruttano que-

sto genere di modello matematico per ottenere una soluzione non necessariamente ottima del problema trattato. In particolare Proximity Search va a modificare il modello che non sarà più riferito alla soluzione ottima del problema ma ad una soluzione vicina alla migliore disponibile ed a costo minore. Ciò che si vuole ottenere con questo euristico è un miglioramento della soluzione di partenza attraverso la risoluzione di più problemi MIP-based in serie, utilizzando per esempio CPLEX come MIP solver.

2.2 Principio di funzionamento

Dato un problema MIP, l'algoritmo Proximity Search modifica la funzione obiettivo minimizzando non più il costo della soluzione ma la distanza dalla migliore soluzione disponibile; inoltre viene aggiunto un vincolo riguardante il costo della soluzione. La funzione obiettivo viene sostituita con un'altra che ha lo scopo di penalizzare una soluzione x in base alla distanza di Hamming da quella corrente per guidare euristicamente la ricerca verso soluzioni prossime e migliori a quella attuale. Partiamo dal seguente problema generico di programmazione mista intera $0-1$:

$$\begin{cases} min\ c^T x \\ Ax \geq b \\ x_j \in \{0,1\} \qquad \forall\, j \in B \end{cases}$$

dove $B \subseteq N = \{1, ..., n\}$. A partire da una soluzione feasible del problema $\widetilde{x}$, la funzione obiettivo viene sostituita con la distanza di Hamming tra x e $\tilde{x}$:

$$\Delta(x, \widetilde{x}) = \sum_{j \,\epsilon\, B : \tilde{x_j} = 0} x_j + \sum_{j \,\epsilon\, B : \tilde{x_j} = 1} (1 - x_j)$$

Inoltre si aggiunge al modello il seguente vincolo di cutoff:

$$c^T x \leq c^T \widetilde{x} - \theta$$

dove $\theta > 0$ è una tolleranza di cutoff che indica il miglioramento minimo della nuova soluzione cercata.

Ecco lo schema di funzionamento dell'intero algoritmo, partendo da una soluzione feasible $\widetilde{x}$:

- Ripeti:
  1. Aggiungere il vincolo $c^T x \leq c^T \widetilde{x} - \theta$ al modello
  2. Sostituire $c^T x$ con la funzione di prossimità $\Delta(x, \widetilde{x})$
  3. Risolvere il modello con un MIP solver finchè non viene raggiunta una delle condizioni per il termine dell'esecuzione, con x^* nuova soluzione trovata (x^* vuota se non è stata trovata nessuna soluzione)
  4. Se x^* non è vuota e $B \subset N$
     - Raffinare x^* risolvendo il seguente problema lineare:

$$x^* := argmin \left\{ c^T(x) : Ax \geq b, x_j = x_j^* \; \forall j \in B \right\}$$

  5. Reimpostare $\Delta(x, \widetilde{x})$ assegnando $\widetilde{x} := x^*$
- Finchè non viene raggiunta una delle condizioni generali per la fine dell'algoritmo (time limit raggiunto, numero iterazioni massimo raggiunto, x^* vuota)

Nel nostro caso il raffinamento non porta a nuove soluzioni dato che le variabili presenti nella funzione obiettivo $c^T x$ da minimizzare sono tutte intere, quindi fissate ai corrispondenti valori di x^*. Perciò x^* è soluzione del problema lineare di raffinamento, ed essendo utilizzata come soluzione di partenza (MIP-start) del problema, viene restituita senza modifiche come soluzione del raffinamento.

2.3 Implementazione

Nel caso del problema TSPTW si passa dalla formulazione del problema descritta nel capitolo precedente alla nuova formulazione per l'algoritmo Proximity Search cambiando la funzione obiettivo (che diventa la distanza di Hamming dalla migliore soluzione conosciuta) ed aggiungendo un nuovo vincolo sul costo della

nuova soluzione. Per quanto riguarda il valore di θ, questo è impostato ad un valore basso ($\theta = 1$) per velocizzare le singole iterazioni che compongono l'algoritmo. La risoluzione del nuovo problema è stata eseguita con il solver CPLEX, tramite la funzione descritta in [Fis13]

$$void\ proxy(double\ TMAX,\ int\ max\_iter,\ double\ theta,$$
$$CPXLPptr\ plp,\ double^*\ values)$$

Questa funzione riceve in ingresso un generico problema sotto forma di un puntatore ad uno spazio di variabili di CPLEX ($CPXLPptr$), il tempo massimo di esecuzione $TMAX$, il numero massimo di iterazioni max_iter, il valore θ (theta) ed un puntatore $values$ ad un array contenente la soluzione iniziale. Quindi il problema viene trasformato come descritto nella sezione precedente e risolto iterativamente salvando in $values$ la migliore soluzione trovata.

Nella risoluzione del problema con CPLEX l'esecuzione si ferma alla prima soluzione feasible trovata, che sarà anche la migliore soluzione conosciuta fino a questo punto dato il vincolo di cutoff, quindi viene modificata la funzione obiettivo riferendola alla distanza di Hamming dalla nuova soluzione disponibile, e si ripete il procedimento finchè non viene raggiunta una delle possibili condizioni per la conclusione dell'algoritmo (time limit raggiunto, problema infeasible, iterazioni massime raggiunte).

All'interno di $proxy()$ è stato fatto un tuning di alcuni parametri per migliorare l'esecuzione di CPLEX. Ad esempio sono stati modificati i parametri relativi all'esecuzione di RINS ad ogni nodo dell'albero e di Polishing dopo 10 secondi dall'inizio dell'esecuzione del problema. RINS (Relaxation Induced Neighborhood Search) è un euristico che esplora le possibili soluzioni vicine all'incumbent corrente del problema per cercare un nuovo incumbent. L'esplorazione delle soluzioni viene effettuata risolvendo un altro MIP (subMIP) per un numero di nodi limitato. Il polishing è un algoritmo di post-processing di tipo genetico che lavora con una popolazione di soluzioni iniziali (anche infeasible), e cerca una soluzione ammissiblile al problema. Dalle prove effettuate l'euristico polishing è risultato

essere molto performante, per questo viene avviato dopo pochi secondi dall'inizio dell'esecuzione di CPLEX.

Come spiegato in precedenza, Proximity Search ha bisogno di una soluzione ammissibile di partenza. Nel capitolo successivo viene descritto un euristico utilizzato per ottenere le soluzioni ammissibili di partenza per il Proximity Search.

Capitolo 3

Simulated Annealing

In seguito viene descritto il principio fisico sul quale si basa l'algoritmo euristico detto Simulated Annealing, il suo funzionamento generale e l'applicazione al problema del Commesso Viaggiatore con Finestre Temporali.

3.1 Descrizione algoritmo

L'approccio consiste nel cercare soluzioni elaborando soltanto un sottoinsieme dello spazio di ricerca. La ricerca locale si basa sul presupposto che il miglioramento di una soluzione si trovi nelle vicinanze di quest'ultima, ovvero nel suo vicinato (neighborhood). Data una qualsiasi soluzione euristica, con tale tecnica si verifica l'esistenza di soluzioni migliori nell'insieme delle soluzioni più vicine. Iterando più volte il procedimento, tale tecnica riesce a produrre, in un tempo relativamente breve, soluzioni piuttosto buone ma senza garanzia a priori della vicinanza all'ottimo globale.

Grazie a tale caratteristica gli algoritmi euristici di ricerca locale utilizzano una quantità di memoria limitata e si prestano ad essere utilizzati negli spazi di ricerca molto vasti. E' possibile progettare euristiche specifiche per qualsiasi problema di ottimizzazione combinatoria, ma esistono alcuni approcci euristici di tipo più generale, detti meta-euristiche. Spesso queste meta-euristiche traggono ispirazione da alcune analogie con la natura fisica come ad esempio Genetic Algoritm, Simulated

Annealing, Ant Colony Optimization, ecc.. La loro struttura e l'idea di fondo di ciascuna metaeuristica sono sostanzialmente generali, ma l'implementazione delle varie componenti dell'algoritmo e l'individuazione dei parametri ottimali, dipende dai singoli problemi.

Ricercare il minimo globale di una funzione di costo con molti gradi di libertà diventa un problema in quanto si può incappare in situazioni di ottimo locale che impediscono l'individuazione delle soluzioni ottimali. Questo rischio, tuttavia, può essere ridotto perfezionando le tecniche euristiche di ricerca e di localizzazione delle soluzioni candidate. L'algoritmo Simulated Annealing adotta una tecnica stocastica sviluppata originariamente da Kirkpatric [Kirk84], che si ispira alla meccanica statistica per trovare soluzioni per problemi di ottimizzazione sia continua che discreta. Questo meccanismo probabilistico consente alla procedura di ricerca di fuggire dai minimi locali.

La strategia alla base del Simulated Annealing è quella di simulare il comportamento del processo termodinamico di tempra dei metalli (annealing) durante il raffreddamento dallo stato fuso. Un solido, portato a tale stato, mediante riscaldamento ad alte temperature, viene riportato poi di nuovo allo stato solido o cristallino, a temperature basse, controllando e riducendo gradualmente la temperatura. Ad alte temperature, gli atomi nel sistema si trovano in uno stato altamente disordinato dovuto al fatto che l'energia del sistema è elevata. Per portare tali atomi in una configurazione cristallina altamente ordinata (statisticamente), deve essere abbassata la temperatura del sistema. Riduzioni veloci della temperatura possono causare, però, difettosità nel reticolo cristallino con conseguenti fratture del reticolo stesso dovute allo stress termico. L'annealing evita questo fenomeno procedendo ad un graduale raffreddamento, portando il sistema ad una struttura globalmente ottima stabile. Il sistema si dice essere in equilibrio termico alla temperatura T se la probabilità $P(E_i)$ di uno stato avente energia E_i è governata dalla distribuzione di Boltzmann :

$$P\left(E_i\right) = \frac{exp\left(-\frac{E_i}{K_b T}\right)}{\sum_i exp\left(-\frac{E_i}{K_b T}\right)}$$

3.2 Principio di funzionamento

La simulazione del processo di annealing applicato a problemi di ottimizzazione richiede di identificare nel problema di ottimizzazione le analogie con i concetti fisici:

- l'energia diventa la funzione di costo
- le configurazioni di particelle divengono i valori delle variabili del problema da ottimizzare
- ricercare uno stato di minima energia significa ricercare una soluzione che minimizza la funzione di costo
- la temperatura diventa un parametro di controllo

L'idea è quella di accettare, in un primo momento (quando la temperatura è alta), oltre alle transizioni che corrispondono a miglioramenti nella funzione obiettivo, anche quelle transizioni che portano a peggioramenti nel valore di tale funzione. La probabilità di accettare tali deterioramenti varia nel corso del processo di ricerca (man mano che la temperatura diminuisce), e scende lentamente verso zero. Verso la fine della ricerca, quando vengono accettati solo miglioramenti, questo metodo diventa una semplice ricerca. La possibilità di spaziare in punti dello spazio di ricerca che peggiorano la soluzione ottima corrente consente di abbandonare eventuali minimi locali ed esplorare meglio l'insieme delle soluzioni ammissibili.

Il metodo comincia da una configurazione iniziale del sistema con energia E_0. Vengono poi generate successive configurazioni con piccole perturbazioni casuali della configurazione corrente. Viene deciso se accettare o rigettare la configurazione in base alla differenza fra l'energia della configurazione corrente e quella della nuova configurazione candidata. L'algoritmo accetta sempre una soluzione candidata la cui energia E_j è inferiore a quella della configurazione corrente E_i. Per contro se l'energia E_j della configurazione candidata è più grande di quella della configurazione corrente, allora la nuova soluzione è accettata con la seguente

probabilità :

$$P(\Delta E) = exp\left(-\frac{\Delta E}{T}\right)$$

Ad alte temperature, l'algoritmo SA può attraversare quasi tutto lo spazio delle soluzioni poiché le soluzioni peggioranti vengono facilmente accettate. Successivamente, abbassandosi il valore del parametro T, l'algoritmo viene confinato in regioni sempre più ristrette dello spazio delle soluzioni dato che la distribuzione di Boltzmann collassa a più basse probabilità di accettazione con il diminuire della temperatura. Di seguito viene presentato lo pseudocodice corrispondente all'algoritmo che implementa il Simulated Annealing:

1. Generare una soluzione di partenza S e impostare la soluzione iniziale campione a $S^* = S$.
2. Determinare una temperatura di partenza T.
3. Finché il sistema non è freddo (T>1):
   (a) Finché non si è raggiunto l'equilibrio per questa temperatura:
      i. Scegliere un vicino casuale S' della soluzione corrente.
      ii. Definire $\Delta = costo(S') - costo(S)$.
      iii. Se $\Delta \geq 0$ (mossa migliorante)
         - Definire $S = S'$
         - se $costo(S) < costo(S^*)$, impostare $S^* = S$.
      iv. Else (mossa peggiorante)
         - Scegliere un numero casuale r uniformemente distribuito tra $[0, 1]$
         - se $r < e^{-\Delta/T}$, definire $S = S'$.
      v. Fine while loop equilibro
   (b) Abbassare la temperatura
   (c) Fine while temperatura
4. Return S^*.

Ad ogni iterazione l'algoritmo, data una soluzione S corrente, procede nel seguente modo:

- se trova S' migliore di S, la accetta
- altrimenti si accetta una soluzione peggiore S'' con una probabilità via via decrescente

Quindi, si deve scegliere un opportuno schema di annealing consistente nella regolazione dei parametri da cui dipende il processo di ottimizzazione: si tratta cioè di stabilire la legge di decadimento della temperatura e la durata del tempo necessario per il raggiungimento dell'equilibrio termico a ciascuna temperatura. Infine si deve introdurre un metodo di perturbazione del sistema che consenta di esplorare lo spazio di ricerca generando nuove configurazioni.

3.3 Applicazione al TSPTW

Nel caso del problema del commesso viaggiatore con finestre temporali, si parte da una soluzione formata dai nodi ordinati per tempo di chiusura della finestra temporale b_i crescente; questa soluzione probabilmente non sarà feasible ma intuitivamente è un buon punto di partenza per arrivare ad una soluzione ammissibile con l'applicazione del Simulated Annealing.

Nella funzione che applica il Simulated Annealing innanzitutto si procede con l'inizializzazione dei parametri dell'algoritmo come la temperatura di partenza T, il fattore di riduzione della temperatura α e il numero di iterazioni da applicare per raggiungere l'equilibrio ad ogni temperatura β. I valori di questi parametri sono stati settati dopo un tuning basato su un sottoinsieme il più vario possibile dei problemi presi in considerazione in questa tesi. I valori da noi scelti sono $T = |V|$ (numero dei vertici), $\alpha = 0.9$ e $\beta = 100$. Questi valori possono subire delle modifiche durante l'esecuzione dell'algoritmo, modifiche che spiegheremo in seguito.

La generazione delle soluzioni rappresenta un punto cardine sia dal punto di vista del tempo di esecuzione dell'algoritmo sia dal punto di vista dell'esplora-

zione dello spazio delle soluzioni del problema. È necessario che la procedura di generazione, detta anche perturbazione, sia il più veloce possibile. In questa implementazione la perturbazione sarà un semplice scambio fra 2 nodi nella sequenza dei nodi visitati. L'energia di una soluzione viene calcolata come

$$E = \sum max\,(0, t_i - b_i)$$

ricordando che t_i rappresenta l'istante di ingresso nel nodo i riferito alla sequenza di nodi attuale e calcolato rispettando i vincoli descritti nel primo capitolo, mentre b_i è il tempo di chiusura della finestra temporale di i. L'energia di una soluzione non ammissibile sarà maggiore di zero in quanto se una soluzione non è ammissibile, esiste almeno un nodo $i : t_i > b_i$, quindi la sommatoria avrà almeno un termine maggiore di zero, ed essendo tutti i termini positivi o nulli anche la sommatoria totale sarà maggiore di zero.

Con l'algoritmo Simulated Annealing cerchiamo di abbassare l'energia della soluzione iniziale finchè questa non arriva a zero, cioè finchè non viene trovata una soluzione feasible. Se si arriva alla temperatura finale senza aver trovato una soluzione ad energia nulla, l'algoritmo viene rieseguito ripartendo dalla stessa soluzione di partenza ma cambiando i parametri. Per i primi 7 cicli andiamo ad aumentare α secondo la formula $\alpha = \frac{(1+\alpha)}{2}$. In questo modo all'aumentare di α la temperatura calerà più lentamente, dando la possibilità all'algoritmo di esplorare un maggior numero di soluzioni. Dopodichè non potendo più agire su α (dopo 7 iterazioni $\alpha = 0.998$, che in letteratura viene considerato come limite superiore del range ottimale di α) ripartiamo con $\alpha = 0.9$ ma moltiplicando per 2 la temperatura iniziale precedente. Iteriamo questo procedimento finchè non raggiungiamo la temperatura iniziale di $8 \cdot |V|$.

Con l'aumento della temperatura viene incrementata l'energia iniziale del sistema con l'obiettivo di introdurre uno scostamento maggiore dalla soluzione di partenza. Se non è stata ancora trovata una soluzione feasible ripartiamo da una temperatura iniziale di $|V|/2$, ed iteriamo il procedimento appena descritto non più moltiplicando la temperatura iniziale ma dividendola per 2, fino a raggiungere il valore minimo di $|V|/16$. In questo modo tendiamo ad evitare gli scambi

peggiorativi, avendo fin da subito una bassa probabilità di accettare le mosse che aumentano il costo della soluzione data la bassa temperatura iniziale.

Questa diminuzione di temperatura iniziale contrasta con l'iniziale aumento, ma dalle prove effettuate viene riscontrato che circa l'80% delle soluzioni trovate con il Simulated Annealing vengono individuate nei cicli a temperatura iniziale uguale o superiore a $|V|$, mentre il restante 20% viene ottenuto nei cicli con temperatura iniziale inferiore a $|V|$. Inoltre questo 20% è concentrato in 3 delle 8 classi di problemi considerate nel prossimo capitolo, quindi probabilmente l'efficacia della diminuzione della temperatura iniziale è correlata a determinate tipologie di problemi. Una percentuale rilevante come il 20% giustifica questo particolare tuning dei parametri, e consente di trovare una soluzione feasible con l'applicazione del Simulated Annealing in oltre il 90% delle istanze considerate.

Quando viene trovata una soluzione ad energia nulla, questa viene salvata come migliore soluzione attuale S e ne viene calcolato il costo effettivo come migliore costo attuale $C = \sum_{(i,j)\in S} c_{ij}$. Inoltre viene incrementato α al valore 0.998 per rallentare il calo della temperatura. L'algoritmo prosegue e termina al raggiungimento della temperatura finale, ma chiaramente l'energia non si può più abbassare. Se viene trovata un'altra soluzione ad energia nulla si procede con il calcolo del costo; se questo è minore del migliore costo attuale C, vengono aggiornati la migliore soluzione e il suo costo C.

Per velocizzare l'algoritmo viene utilizzata una struttura dati *str* formata da un array di *struct* $Nodo1$. La struttura dati *struct* $Nodo1$ ha 3 campi : *int nodo*, *double tempo_entrata* e *double* $funz\_obiettivo$. Il campo *nodo* contiene il numero del nodo considerato, *tempo_entrata* rappresenta il tempo di arrivo al nodo mentre $funz\_obiettivo$ rappresenta il termine dovuto al nodo indicato nel primo campo per il calcolo dell'energia E. Quando procediamo allo scambio fra le posizioni $h \neq 0$ e $k > h$ scambiamo i valori del campo *nodo*, quindi procediamo al ricalcolo dalla posizione h alla k dei tempi di entrata e dei termini della funzione obiettivo. Poi iteriamo il ricalcolo del tempo di entrata e della funzione obiettivo per tutte le posizioni oltre k per le quali il nuovo tempo di entrata calcolato è diverso da quello precedente salvato nella struttura *str*.

Inoltre il campo $funz\_obiettivo$ viene utilizzato per velocizzare il calcolo dell'energia E della nuova soluzione dato il valore E' della soluzione precedente. Infatti per tutte le posizioni i nelle quali viene ricalcolato il termine della funzione obiettivo è sufficiente sottrarre ad E' il valore $funz\_obiettivo$ salvato ed aggiungervi quello riferito alla nuova soluzione per ottenere E.

Capitolo 4

Risultati computazionali

In questo capitolo riportiamo i risultati numerici relativi all'applicazione degli algoritmi Simulated Annealing e Proximity Search, per poi valutarne l'efficacia confrontandone le prestazioni con altri metodi di risoluzione esposti in letteratura riguardanti il TSP con Finestre Temporali.

Le istanze considerate sono 602 e si dividono in 8 classi. Le prime 4 classi presentate (Dumas, Gendreau, Ohlmann e Thomas, Urrutia) sono simili nella struttura dei problemi in quanto le ultime 3 classi elencate sono derivate dalla Dumas attraverso l'aumento dei nodi e la variazione della dimensione media delle finestre temporali. Inoltre in tutte queste classi le istanze sono divise in sottoclassi da 5 istanze ciascuna. Le sottoclassi vengono chiamate nxwy, dove x indica il numero di nodi e y la dimensione media delle finestre temporali; le istanze all'interno della sottoclasse vengono indicate con un suffisso che va da .001 a .005. Per queste classi i valori riportati si riferiscono alle medie sulle sottoclassi, mentre per tutte le altre classi i valori riportati riguardano le singole istanze. Anche la classe Langevin è formata da sottoclassi composte da 10 istanze uniformi per numero di nodi e dimensione media delle finestre temporali. La classe Ascheuer è l'unica formata da istanze asimmetriche e derivante da problemi reali, mentre le ultime 2 classi (Potvin - Bengio e Pesant) derivano entrambe da dei tour ricavati da problemi di Vehicle Routing Problem proposti da Solomon [Sol87].

Le istanze provengono da questi due siti web:

- Benchmark Instances for the Travelling Salesman Problem with Time Windows (TSPTW). Sito curato da Manuel López-Ibáñez e Christian Blum che riporta tutti i problemi di 7 delle 8 classi qui utilizzate
  $http://iridia.ulb.ac.be/\ manuel/tsptw-instances$

- The Traveling Salesman Problem with Time Windows (TSPTW) - Approaches & Additional Resources Pagina web relativa al TSPTW curata da Rodrigo Ferreira da Silva and Sebastián Urrutia, nella quale vengono riportati i problemi di test utilizzati per valutare le prestazioni dell'euristico VNS (Variable Neighbourhood Search) proposto proprio da Da Silva - Urrutia [DaS10]. Da qui vengono scaricate le istanze della classe proposta da questi due autori.
  $http://homepages.dcc.ufmg.br/\ rfsilva/tsptw/$

4.1 Test Simulated Annealing + Proximity Search

In questo paragrafo presentiamo i risultati computazionali dei test su istanze TSPTW dell'algoritmo formato da Simulated Annealing e Proximity Search. L'algoritmo inizia con l'esecuzione del Simulated Annealing. Quindi si procede con il preprocessing ed infine si passa al Proximity Search. Il preprocessing viene eseguito dopo il Simulated Annealing in quanto quest'ultimo prevede la possibilità di formare durante l'esecuzione dell'algoritmo delle soluzioni infeasible attraverso scambi casuali di nodi, quindi potrebbero essere utilizzati degli archi eliminati con il preprocessing. L'algoritmo Proximity Search, come spiegato in precedenza, ha bisogno di una soluzione ammissibile di partenza che viene ricavata proprio con il Simulated Annealing. Se l'esecuzione del Simulated Annealing non trova nessuna soluzione feasible, all'interno della funzione $proxy()$ si procede con la risoluzione tramite CPLEX del modello TSPTW con indicator constraints descritto nel capitolo 1 con time limit di un'ora. L'esecuzione in questo caso si ferma alla prima soluzione feasible trovata, che diventa il punto di partenza per il Proximity

Search. In questo caso il tempo utilizzato per trovare una soluzione feasible tramite CPLEX viene sottratto al tempo totale di esecuzione del Proximity Search ed aggiunto al tempo di esecuzione della prima parte dell'algoritmo dedicata alla ricerca di una soluzione ammissibile, che verrà indicata come SA/CPLEX. In ogni caso i tempi delle due fasi dell'algoritmo andranno sommati per il confronto dell'algoritmo SA+PS con altri metodi di risoluzione del TSPTW.

Sottolineamo che non viene fatta alcuna differenza fra le esecuzioni del Proximity Search che terminamo per time limit raggiunto e quelle concluse per problema infeasible, il che significa che non esistono soluzioni che migliorano il costo della soluzione attuale di una quantità almeno pari a θ. Questo per due motivi. Il primo è che il Proximity Search viene pensato come un euristico in grado di migliorare una soluzione di partenza, quindi un algoritmo utile per problemi difficili da risolvere all'ottimo, ed in quest'ottica è ininfluente il motivo di terminazione dell'algoritmo. Il secondo motivo è che alcune classi di problemi hanno dei costi non interi, quindi per queste classi la terminazione del Proximity Search per problema infeasible non significa che sia stata trovata la soluzione ottima al problema, ma semplicemente che l'ottimo ha un costo non più distante di $\theta = 1$ dalla soluzione corrente.

Su 602 istanze totali, per 56 di queste l'algoritmo Simulated Annealing non è stato in grado di trovare una soluzione ammissibile. Considerando queste 56 istanze, per 54 di esse l'esecuzione di CPLEX è efficace nella ricerca di una soluzione feasible, mentre in 2 casi ($rc\_204.1$ e $rc208.0$) nè il Simulated Annealing nè CPLEX riescono a trovare una soluzione feasible. Le istanze non risolte dal Simulated Annealing appartengono solamente a 3 delle 8 classi (Potvin e Bengio, Pesant e Langevin), ed in particolare 46 di queste alla classe di problemi Langevin, per la quale evidentemente l'algoritmo Simulated Annealing non è efficace. Le tabelle riportano i seguenti dati:

- $numVertici$: numero di nodi del problema, quando non è chiaro dal nome dell'istanza

- $valueSA$: migliore soluzione ottenuta tramite il Simulated Annealing; nei casi in cui non viene trovata nessuna soluzione feasible e si ricorre alla ricerca

di una soluzione tramite CPLEX, viene riportata la soluzione trovata da CPLEX se esiste.

- $timeSA$: tempo di esecuzione (CPU time) del Simulated Annealing. Nel caso in cui il Simulated Annealing non trovi alcuna soluzione feasible e quindi si proceda con la ricerca tramite CPLEX, vengono indicati separatamente sia il tempo richiesto dal Simulated Annealing, sia il tempo di esecuzione di CPLEX in quest'ordine.
- $\%elim.$: percentuale di archi eliminati tramite il preprocessing
- $iterPS$: numero di iterazioni concluse durante l'esecuzione di Proximity Search
- $valuePS$: costo della migliore soluzione ottenuta tramite Proximity Search
- $timePS$: tempo, inteso come CPU time, dall'inizio dell'esecuzione di Proximity Search alla fine dell'ultima iterazione conclusa con successo, cioè non per time limit.
- $\%migl.$: percentuale di miglioramento della soluzione finale ottenuta dal Proximity Search rispetto a quella di partenza (soluzione Simulated Annealing o CPLEX).

Tutte le prove sono state effettuate presso il cluster di calcolo Blade, composto da 14 'lame' di calcolo modello DELL PowerEdge M600 ciascuna equipaggiata con:

- 2 Processori quad core Intel Xeon E5450 (12MB Cache, 3.00 GHz)
- 16 GB RAM
- 2 Hard disk da 72GB in configurazione RAID-1 (mirroring)

4.1.1 Istanze proposte da Dumas

27 classi di 5 istanze ciascuna. Tutte le istanze sono state proposte e risolte all'ottimo da Dumas e al. [Dum95]. La taglia delle istanze varia da 20 a 200 nodi, mentre la dimensione media delle finestre temporali va dai 20 ai 100 istanti temporali. Per questa come per le seguenti 4 classi di istanze possiamo notare che un effetto dell'aumento dell'estensione media delle finestre temporali in problemi aventi lo stesso numero di nodi è la diminuzione della percentuale di archi eliminati. A sua volta questa diminuzione comporta un aumento della complessità del problema dato il maggior numero di variabili x_{ij} presenti, sia nel modello originale del TSPTW che in quello risolto da Proximity Search, come si può immediatamente notare dai dati ottenuti. Proximity Search migliora la soluzione ottenuta dal Simulated Annealing del 5.54% in media, del 10% circa per le sottoclassi di problemi più difficili (n150w40, n150w60 ed n200w40). Per quanto riguarda i tempi, la media per il Simulated Annealing (154.96) è molto vicina a quella del Proximity Search (146.27).

Tabella 4.1: Istanze proposte da Dumas

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
n20w20	361.20	1.82	84.51	1.00	361.20	0.01	0.00
n20w40	316.00	1.76	74.67	1.00	316.00	0.11	0.00
n20w60	309.80	1.70	64.35	1.00	309.80	0.66	0.00
n20w80	311.00	1.69	56.67	1.00	311.00	0.25	0.00
n20w100	279.60	1.70	42.95	2.40	275.20	1.76	1.57
n40w20	486.60	4.40	89.06	1.00	486.60	0.14	0.00
n40w40	464.80	4.72	81.42	2.60	461.00	2.36	0.82
n40w60	431.60	4.80	70.75	6.20	416.40	12.34	3.52
n40w80	418.80	4.93	65.69	7.60	399.80	9.71	4.54
n40w100	404.60	4.93	56.33	11.00	377.00	78.82	6.82
n60w20	584.80	7.29	91.68	2.40	581.60	1.36	0.55
n60w40	606.40	7.14	84.48	8.60	590.20	12.53	2.67
n60w60	591.20	6.08	78.68	12.60	560.00	23.45	5.28
n60w80	547.20	7.18	69.81	13.60	508.00	81.50	7.16
n60w100	562.40	5.95	63.62	17.00	514.80	893.47	8.46
n80w20	689.60	105.29	92.60	5.80	676.60	3.73	1.89
n80w40	664.00	7.55	86.15	12.20	630.00	15.71	5.12
n80w60	659.20	14.54	79.83	21.20	607.60	74.28	7.83
n80w80	654.60	13.37	74.64	24.20	595.40	397.90	9.04
n100w20	775.60	14.81	93.54	9.00	757.60	11.90	2.32
n100w40	760.00	34.01	87.85	23.60	701.80	35.15	7.66
n100w60	755.80	109.29	80.97	22.80	698.80	330.05	7.54
n150w20	915.00	304.21	94.75	23.40	868.40	27.24	5.09
n150w40	920.60	71.44	89.42	36.40	834.80	238.59	9.32
n150w60	925.40	477.77	85.62	41.80	820.00	602.44	11.39
n200w20	1082.60	957.46	95.19	35.40	1009.80	194.35	6.72
n200w40	1093.80	2007.99	90.78	49.40	984.20	899.42	10.02
MEDIA	613.79	154.96	78.74	14.60	579.76	146.27	5.54

4.1.2 Istanze proposte da Gendreau

Le istanze proposte da Gendreau [Gen98] sono 140 raggruppate in 28 classi uniformi per numero di nodi e ampiezza media delle finestre temporali. Le istanze sono state ricavate dai problemi proposti da Dumas [Dum95] esclusi quelli con più di 100 nodi, estendendo le finestre temporali fino a 200 unità temporali di media. Intuitivamente l'effetto di questa estensione delle finestre temporali, oltre alla diminuzione degli archi eliminati dal preprocessing, è l'aumento dei tour che rispettano i limiti delle finestre temporali, cioè dei tour feasible. Per questo la soluzione trovata con il Simulated Annealing probabilmente sarà più distante dall'ottimo, e infatti confrontando i risultati ottenuti per le istanze delle due classi con uguale numero di nodi e tempi di percorrenza osserviamo un netto aumento dei tempi del Proximity Search con il sostanziale raddoppio delle iterazioni medie e di conseguenza l'aumento della percentuale di miglioramento della soluzione (17.02% Gendreau, 5.54% Dumas)

Tabella 4.2: Istanze proposte da Gendreau

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
n20w120	274.00	1.67	31.38	3.00	265.60	8.27	3.07
n20w140	243.80	1.66	24.00	5.40	232.80	5.82	4.51
n20w160	231.40	1.63	22.29	5.20	218.20	4.40	5.70
n20w180	258.40	1.66	16.86	7.00	236.60	109.00	8.44
n20w200	272.00	1.61	11.38	8.60	241.00	229.98	11.40
n40w120	427.80	6.13	46.38	13.00	377.80	72.17	11.69
n40w140	427.20	6.05	40.71	15.40	364.40	709.11	14.70
n40w160	393.60	6.06	31.62	16.60	326.80	943.45	16.97
n40w180	399.40	5.88	27.98	19.40	334.40	666.43	16.27
n40w200	392.00	6.22	20.34	21.60	314.40	365.71	19.80
n60w120	534.40	11.83	52.91	23.40	451.00	981.03	15.61
n60w140	539.40	14.48	49.34	26.00	453.20	1122.30	15.98
n60w160	577.40	11.19	43.14	28.80	466.00	76.92	19.29
n60w180	540.20	10.76	34.55	26.60	428.20	368.61	20.73
n60w200	543.00	10.52	30.61	29.80	430.20	494.45	20.77
n80w100	660.00	9.71	66.34	25.40	580.00	1192.04	12.12
n80w120	657.60	17.58	56.30	37.40	544.00	570.00	17.27
n80w140	644.00	17.41	52.40	34.80	507.40	553.14	21.21
n80w160	639.40	15.46	45.60	41.40	506.20	1294.00	20.83
n80w180	643.20	18.32	42.18	36.40	503.20	2603.37	21.77
n80w200	638.20	19.26	34.52	40.80	485.20	1727.86	23.97
n100w80	757.00	21.81	75.94	31.80	666.40	343.17	11.97
n100w100	758.20	16.50	69.57	38.40	643.40	726.34	15.14
n100w120	797.40	26.12	63.71	47.60	600.40	864.68	24.71
n100w140	772.80	24.91	59.57	56.20	548.80	673.88	28.99
n100w160	772.80	24.62	56.27	62.20	556.40	2413.47	28.00
n100w180	731.40	20.51	45.61	46.40	570.20	2373.81	22.04
n100w200	740.20	28.43	40.02	51.60	566.40	2397.94	23.48
MEDIA	545.22	12.79	42.55	28.58	443.52	853.26	17.02

4.1.3 Istanze proposte da Ohlmann e Thomas

25 istanze raggruppate in 5 sottoclassi proposte per la prima volta da Ohlmann e Thomas [Ohlm07]. Le istanze sono state ricavati dai problemi con 150 e 200 nodi proposti da Dumas [Dum95], aumentando la dimensione delle finestre temporali di 100 unità di tempo. Anche in questa classe notiamo le stesse cose osservate per la precedente, cioè un aumento del tempo riferito al Proximity Search, con una media (3102.36) vicina al time limit, ed una percentuale di miglioramento molto alta, in media superiore al 20%.

Tabella 4.3: Istanze proposte da Ohlmann e Thomas

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
n150w120	903.40	61.96	65.69	59.20	728.80	2366.69	19.33
n150w140	911.40	32.45	60.88	67.60	711.00	3181.99	21.99
n150w160	888.80	38.31	57.85	67.20	680.40	3189.08	23.45
n200w120	1050.40	67.64	69.23	80.40	821.80	3281.47	21.76
n200w140	1049.20	131.46	65.60	80.40	824.40	3492.55	21.43
MEDIA	960.64	66.36	63.85	70.96	753.28	3102.36	21.59

4.1.4 Istanze proposte da Urrutia

Queste istanze vengono proposte per la prima volta da Da Silva ed Urrutia [DaS10]. Sono divise in 25 sottoclassi da 5 istanze ciascuna, e si caratterizzano per l'elevato numero di nodi (da 200 a 400) e la grande ampiezza media delle finestre temporali (da 100 a 500 unità temporali). Per queste istanze il Simulated Annealing impiega in media 93.72 secondi. La media dei tempi del Proximity Search è di 1158.76, anche se per questa classe i tempi sono molto variabili in base alla difficoltà del problema. In ogni caso Proximity Search risulta efficace, migliorando la soluzione di partenza in media dell' 11.99%.

Tabella 4.4: Istanze proposte da Urrutia

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
n200w100	10037.80	64.70	98.46	6.20	10019.60	2.59	0.18
n200w200	9570.60	36.78	97.60	25.20	9252.00	20.74	3.33
n200w300	8934.80	43.37	96.46	65.80	8022.80	605.46	10.21
n200w400	8529.40	59.26	95.12	92.20	7079.20	2409.04	17.00
n200w500	8313.20	51.09	94.10	103.40	6484.40	1115.30	22.00
n250w100	12667.60	60.95	98.70	8.80	12633.00	3.84	0.27
n250w200	11830.00	36.88	98.03	39.00	11310.40	42.59	4.39
n250w300	11389.60	62.83	97.27	84.40	10230.40	242.30	10.18
n250w400	10904.20	57.58	96.19	123.80	8900.40	1562.72	18.38
n250w500	10702.20	69.83	95.25	147.40	8123.40	2324.01	24.10
n300w100	15090.20	139.98	98.95	8.80	15041.20	42.01	0.32
n300w200	14448.80	83.78	98.43	46.80	13851.40	40.34	4.13
n300w300	13125.60	81.98	97.55	111.80	11479.40	965.77	12.54
n300w400	12897.00	119.44	96.78	152.00	10427.60	2248.84	19.15
n300w500	13118.40	97.94	96.08	188.20	10001.60	3068.53	23.76
n350w100	17580.20	186.33	99.10	12.80	17494.00	60.20	0.49
n350w200	16526.60	124.65	98.58	66.20	15672.00	494.06	5.17
n350w300	15720.80	47.99	97.95	117.40	13648.60	1035.35	13.18
n350w400	15216.40	124.11	97.20	179.80	12157.80	2259.33	20.10
n350w500	15404.40	78.23	96.63	215.20	11484.00	3031.60	25.45
n400w100	19552.80	261.06	99.22	17.00	19454.80	7.77	0.50
n400w200	19485.00	72.27	98.78	74.20	18439.80	75.60	5.36
n400w300	18437.60	83.62	98.23	135.60	15871.80	601.22	13.92
n400w400	17667.00	119.11	97.60	213.20	14110.80	3191.98	20.13
n400w500	17276.20	179.32	96.97	267.20	12850.40	3517.92	25.62
MEDIA	13777.06	93.72	97.41	100.10	12161.63	1158.76	11.99

4.1.5 Istanze proposte da Langevin

70 istanze divise in sottoclassi di 10 istanze proposte e risolte all'ottimo da Langevin [Lang93]. Le istanze hanno dai 20 ai 60 nodi, mentre la media delle finestre temporali va da 20 a 40 unità temporali. Le istanze originali proposte da Langevin sarebbero 90, cioè 10 per ogni combinazione di 20, 40 e 60 nodi e 20, 30, 40 unità di larghezza massima delle finestre temporali. Tuttavia due di queste sottoclassi (n20ft20 ed n40ft30) non vengono citate nella letteratura successiva all'articolo di Langevin [Lang93] e non sono disponibili nei due siti web sopra citati. Per questa classe in più della metà dei casi il Simulated Annealing risulta inefficace, ed il valore riportato proviene dalla risoluzione del modello matematico TSPTW con CPLEX. In questi casi il tempo di esecuzione della prima parte dell'algoritmo sarà composto da un primo termine dovuto all'esecuzione del Simulated Annealing, e da un secondo termine che rappresenta il tempo di esecuzione di CPLEX. A differenza delle prime quattro classi vengono riportati i valori relativi alle singole istanze oltre alle medie delle sottoclassi. Tuttavia per il calcolo della percentuale di miglioramento relativa al Proximity Search e delle medie finali relative alla classe Langevin vengono considerate solamente le sottoclassi per uniformità con le prime quattro classi. Per questa classe di matrici l'algoritmo Proximity Search non porta ad un miglioramento delle soluzioni apprezzabile (0.91% medio). Questo è dovuto al fatto che in più della metà dei casi utilizziamo CPLEX per la ricerca di una soluzione feasible, e per questo tipo di problemi con pochi nodi CPLEX risulta molto efficace trovando una soluzione vicina all'ottimo in pochissimo tempo.

Tabella 4.5: Istanze proposte da Langevin - 20/40 nodi

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl
N20ft301	661.60[a]	30.65+0.01	89.00	1.00	661.60	0.01	
N20ft302	713.30	0.03	87.47	3.00	684.20	0.67	
N20ft303	746.40	0.03	88.83	1.00	746.40	0.01	
N20ft304	817.00	0.02	88.22	1.00	817.00	0.01	
N20ft305	728.50	0.03	87.18	2.00	716.50	0.01	
N20ft306	733.50	0.02	87.14	2.00	727.80	0.01	
N20ft307	691.80	33.65	85.97	1.00	691.80	0.01	
N20ft308	788.20	0.02	87.47	1.00	788.20	0.01	
N20ft309	757.90	0.03	83.73	3.00	730.70	0.52	
N20ft310	683.00	32.14	86.09	1.00	683.00	0.01	
n20ft30	732.12	9.66	87.11	1.60	724.70	0.13	1.01
N20ft401	660.80	34.53	86.75	1.00	660.80	0.01	
N20ft402	713.30	0.03	85.08	4.00	684.20	0.02	
N20ft403	746.40	0.04	87.76	1.00	746.40	0.01	
N20ft404	817.00	0.03	86.88	1.00	817.00	0.01	
N20ft405	716.50[a]	30.42+0.01	84.42	1.00	716.50	0.01	
N20ft406	733.50	0.04	85.04	2.00	727.80	0.01	
N20ft407	698.70	0.03	83.90	2.00	691.80	1.21	
N20ft408	757.30	31.76	86.35	1.00	757.30	0.01	
N20ft409	749.20	27.59	81.89	2.00	730.70	0.03	
N20ft410	683.00	10.52	82.11	1.00	683.00	0.01	
n20ft40	727.57	13.50	85.02	1.60	721.50	0.13	0.83
N40ft201	1112.00	0.13	94.29	2.00	1100.60	0.01	
N40ft202	1010.40[a]	117.43+0.01	94.44	1.00	1010.40	0.01	
N40ft203	876.80[a]	115.11+0.01	93.52	1.00	876.80	0.01	
N40ft204	898.90[a]	116.82+0.01	93.45	2.00	885.80	0.02	
N40ft205	973.30	0.12	93.05	4.00	940.90	0.99	
N40ft206	1054.20	107.92	94.01	1.00	1054.20	0.01	
N40ft207	887.90	0.12	93.45	2.00	868.10	0.72	
N40ft208	1050.70[a]	117.45+0.02	93.38	1.00	1050.70	0.03	
N40ft209	1023.20[a]	116.68+0.01	93.30	2.00	1013.90	0.02	
N40ft210	1037.90[a]	116.69+0.01	94.40	2.00	1026.30	0.01	
n40ft20	992.53	80.85	93.73	1.80	982.70	0.18	0.99
N40ft401	1085.00[a]	115.37+0.05	90.60	1.00	1085.00	0.36	
N40ft402	1019.20[a]	116.29+0.01	92.21	4.00	995.60	1.60	
N40ft403	853.90[a]	114.78+0.02	87.72	3.00	845.80	1.58	
N40ft404	868.00[a]	116.31+0.17	88.79	1.00	868.00	0.20	
N40ft405	994.40[a]	115.51+0.12	88.73	3.00	936.50	1.22	
N40ft406	969.10[a]	115.54+0.12	89.07	1.00	969.10	0.57	
N40ft407	831.20[a]	115.67+0.06	89.16	1.00	831.20	0.13	
N40ft408	1038.50	116.59	89.24	7.00	1002.70	1.95	
N40ft409	1000.50[a]	118.70+0.09	89.96	1.00	1000.50	0.14	
N40ft410	997.20	57.21	89.32	3.00	983.80	0.06	
n40ft40	965.70	110.26	89.48	2.50	951.80	0.78	1.44

[a] valore trovato con CPLEX

Tabella 4.6: Istanze proposte da Langevin - 60 nodi

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
N60ft201	1363.50[a]	265.64+0.01	95.21	3.00	1353.50	0.32	
N60ft202	1161.60[a]	261.68+0.08	95.26	1.00	1161.60	0.09	
N60ft203	1182.90[a]	263.98+0.04	95.04	1.00	1181.90	0.28	
N60ft204	1265.50[a]	264.30+0.08	95.21	3.00	1257.50	0.12	
N60ft205	1185.10[a]	262.39+0.01	95.23	2.00	1184.10	0.02	
N60ft206	1215.20[a]	260.12+0.01	94.80	4.00	1199.60	1.48	
N60ft207	1307.10[a]	268.50+0.01	95.80	4.00	1299.00	0.05	
N60ft208	1144.30[a]	261.94+0.01	94.65	3.00	1113.00	1.56	
N60ft209	1231.00[a]	260.52+0.01	94.82	7.00	1171.30	2.71	
N60ft210	1234.30[a]	264.88+0.08	94.51	1.00	1234.30	0.86	
n60ft20	1229.05	263.43	95.05	2.90	1215.70	0.75	1.09
N60ft301	1339.60[a]	267.17+0.12	93.54	2.00	1337.00	0.16	
N60ft302	1089.50[a]	259.16+0.13	93.69	1.00	1089.50	0.52	
N60ft303	1179.00[a]	273.37+0.12	93.32	1.00	1179.00	0.28	
N60ft304	1230.20[a]	267.57+0.10	92.83	2.00	1230.00	1.02	
N60ft305	1162.40[a]	261.64+0.04	93.30	5.00	1151.60	1.94	
N60ft306	1173.80[a]	258.83+0.11	92.27	3.00	1167.90	2.31	
N60ft307	1220.10[a]	267.50+0.06	93.68	1.00	1220.10	0.11	
N60ft308	1110.70[a]	259.14+0.05	92.77	5.00	1097.60	1.93	
N60ft309	1207.90[a]	263.11+0.01	93.23	8.00	1140.60	3.54	
N60ft310	1225.20[a]	263.43+0.06	92.44	4.00	1219.20	2.49	
n60ft30	1193.84	264.17	93.11	3.20	1183.20	1.43	0.89
N60ft401	1335.00[a]	276.18+0.10	91.80	1.00	1335.00	0.36	
N60ft402	1107.90[a]	261.26+0.06	92.21	5.00	1088.10	1.51	
N60ft403	1178.90[a]	262.43+0.15	91.02	4.00	1173.70	3.84	
N60ft404	1184.90[a]	272.35+0.07	91.41	1.00	1184.90	0.33	
N60ft405	1154.20[a]	261.05+0.14	91.14	3.00	1146.20	2.33	
N60ft406	1152.00[a]	256.24+0.17	89.44	7.00	1140.20	5.92	
N60ft407	1211.60[a]	262.63+0.14	91.76	6.00	1198.90	4.16	
N60ft408	1029.40[a]	261.69+0.09	90.58	1.00	1029.40	0.84	
N60ft409	1157.40[a]	262.14+0.06	91.31	6.00	1121.50	2.13	
N60ft410	1198.70[a]	268.15+0.68	90.31	3.00	1189.60	8.23	
n60ft40	1171.00	264.58	91.10	3.70	1160.80	2.97	0.87
MEDIA[b]	1001.69	143.78	90.66	2.47	991.49	0.91	1.02

[a] valore trovato con CPLEX

[b] media dei valori di tutte le sottoclassi che compongono la classe Langevin

4.1.6 Istanze proposte da Ascheuer

50 istanze asimmetriche proposte da Ascheuer, Fischetti e Grötschel [Asch01] che rappresentano casi reali derivati da uno studio industriale con l'intento di ridurre al minimo il tempo di viaggio di un carico in un magazzino automatizzato. Il numero di nodi è ricavabile dal numero che segue rbg nel nome dell'istanza. Nella maggior parte dei casi, la matrice rbg$x$ avrà $x+2$ nodi, tranne nei casi delle serie rbg17 e rbg21, dove i nodi sono rispettivamente 17 e 21. Per questa classe il miglioramento medio della soluzione dovuto al Proximity Search è del 3.22%.

Tabella 4.7: Istanze proposte da Ascheuer

	SA			PS			
nome	value	time	%elim.	iter	value	time	%migl.
rbg010a	671.00	0.33	37.27	1.00	671.00	1.58	0.00
rbg016a	938.00	0.89	62.13	1.00	938.00	0.03	0.00
rbg016b	1305.00	1.09	30.88	2.00	1304.00	5.09	0.08
rbg017	893.00	0.79	38.33	1.00	893.00	0.77	0.00
rbg017.2	874.00	0.72	10.42	4.00	852.00	3.20	2.52
rbg017a	4296.00	1.32	13.73	1.00	4296.00	1.46	0.00
rbg019a	1262.00	1.29	71.84	1.00	1262.00	0.01	0.00
rbg019b	1870.00	1.15	36.84	3.00	1866.00	6.45	0.21
rbg019c	4536.00	1.63	16.05	1.00	4536.00	21.64	0.00
rbg019d	1356.00	1.32	51.32	1.00	1356.00	0.04	0.00
rbg020a	4689.00	1.80	22.62	1.00	4689.00	1.10	0.00
rbg021	4536.00	1.62	16.05	1.00	4536.00	22.17	0.00
rbg021.2	4559.00	1.63	14.47	9.00	4528.00	12.68	0.68
rbg021.3	4560.00	1.47	14.21	11.00	4528.00	52.28	0.70
rbg021.4	4559.00	1.39	12.89	7.00	4525.00	22.32	0.75
rbg021.5	4551.00	1.36	12.37	7.00	4515.00	6.81	0.79
rbg021.6	4532.00	1.31	3.16	5.00	4480.00	4.68	1.15
rbg021.7	4523.00	1.17	0.53	13.00	4479.00	12.77	0.97
rbg021.8	4539.00	1.12	0.00	8.00	4478.00	5.20	1.34
rbg021.9	4538.00	0.97	0.00	16.00	4478.00	15.50	1.32
rbg027a	5094.00	2.06	17.99	2.00	5091.00	18.51	0.06
rbg031a	1920.00	3.02	55.04	12.00	1863.00	12.41	2.97
rbg033a	2099.00	3.36	57.75	6.00	2069.00	12.25	1.43
rbg034a	2264.00	4.29	49.83	7.00	2222.00	12.41	1.86
rbg035a	2280.00	4.66	57.06	15.00	2114.00	14.19	7.28
rbg035a.2	2282.00	3.74	22.14	39.00	2056.00	113.30	9.90
rbg038a	2571.00	5.08	61.13	9.00	2480.00	26.47	3.54
rbg040a	2523.00	5.43	62.07	23.00	2378.00	2159.18	5.75
rbg041a	2731.00	5.62	59.18	20.00	2598.00	181.47	4.87
rbg042a	2908.00	6.32	53.77	24.00	2774.00	577.06	4.61
rbg048a	9518.00	5.62	32.40	27.00	9428.00	667.47	0.95
rbg049a	10187.00	7.01	48.94	26.00	10050.00	2029.18	1.34
rbg050a	3216.00	6.30	32.20	35.00	2953.00	1684.18	8.18
rbg050b	10016.00	7.98	47.61	28.00	9889.00	1930.58	1.27
rbg050c	10153.00	8.77	36.04	27.00	10033.00	759.31	1.18
rbg055a	3832.00	10.41	71.85	16.00	3761.00	26.94	1.85
rbg067a	4713.00	14.93	78.71	20.00	4625.00	54.91	1.87
rbg086a	8651.00	21.21	84.92	34.00	8400.00	199.25	2.90
rbg092a	7473.00	27.64	81.69	54.00	7158.00	1386.91	4.22
rbg125a	8552.00	39.04	86.78	61.00	7936.00	170.62	7.20
rbg132	9158.00	54.17	88.98	74.00	8468.00	457.35	7.53
rbg132.2	9099.00	43.50	80.13	107.00	8211.00	1693.68	9.76
rbg152	10636.00	49.78	89.09	66.00	10032.00	302.11	5.68
rbg152.3	10653.00	70.49	71.39	93.00	9788.00	1818.28	8.12
rbg172a	11909.00	66.40	89.20	88.00	10950.00	2500.95	8.05
rbg193	13231.00	74.50	90.55	94.00	12535.00	3231.02	5.26
rbg193.2	13223.00	96.06	82.47	124.00	12164.00	3370.97	8.01
rbg201a	14121.00	79.35	90.79	135.00	12994.00	2465.98	7.98
rbg233	16173.00	104.47	92.00	119.00	15002.00	2424.11	7.24
rbg233.2	16162.00	134.59	84.88	182.00	14591.00	3485.30	9.72
MEDIA	5818.70	19.80	48.43	33.22	5556.46	679.64	3.22

4.1.7 Istanze proposte da Potvin e Bengio

30 istanze proposte da Potvin e Bengio [Pot96] derivate dalle istanze VRPTW (Vehicle Routing Problem with Time Windows) presentate da Solomon [Sol87]. Queste istanze sono molto diverse fra loro per struttura, e vanno dai 3 ai 44 nodi. Il numero di nodi, non essendo ricavabile dal nome della matrice, viene riportato in tabella. Per la matrice rc_204.1 non viene riportato nessun valore in quanto nè il Simulated Annealing, nè CPLEX riescono a trovare una soluzione ammissibile, quindi non è possibile partire con l'esecuzione di Proximity Search. Non avendo valori questa matrice viene esclusa dal calcolo delle medie dei valori per la classe. La percentuale di miglioramento relativa al Proximity Search per questa classe è in media del 13.55%.

Tabella 4.8: Istanze proposte da Potvin e Bengio

		SA			PS			
nome	nodi	value	time	%elim.	iter	value	time	%migl.
rc_201.1	20	505.26	0.02	52.63	8.00	444.54	5.44	12.02
rc_201.2	26	764.03	0.78	65.38	11.00	711.54	4.22	6.87
rc_201.3	32	794.52[a]	72.84+1.22	69.56	5.00	789.49	2.97	0.63
rc_201.4	26	851.30	37.60	66.92	7.00	793.64	7.52	6.77
rc_202.1	33	886.87	74.31	36.93	21.00	771.78	2770.89	12.98
rc_202.2	14	360.28	0.01	4.95	10.00	304.14	12.04	15.58
rc_202.3	29	907.55	8.96	59.48	17.00	839.58	14.59	7.49
rc_202.4	28	916.47	3.82	32.80	12.00	793.03	49.08	13.47
rc_203.1	19	524.61	1.31	16.96	15.00	453.48	28.13	13.56
rc_203.2	33	923.41	9.49	18.84	17.00	784.16	52.29	15.08
rc_203.3	37	941.98	46.56	18.24	17.00	821.21	311.97	12.82
rc_203.4	15	386.40	0.73	2.86	7.00	314.29	4.34	18.66
rc_204.1	46	—[b]	—	—	—	—	—	—
rc_204.2	33	918.51	4.09	3.03	31.00	662.16	284.77	27.91
rc_204.3	24	646.44	1.60	0.91	30.00	460.47	687.62	28.77
rc_205.1	14	415.14	0.01	39.01	11.00	343.21	7.08	17.33
rc_205.2	27	785.77[a]	52.09+1.64	54.56	4.00	754.29	22.49	4.01
rc_205.3	35	939.24	77.55	37.08	19.00	825.06	20.78	12.16
rc_205.4	28	799.04[a]	57.86+21.14	58.31	12.00	739.33	1624.92	7.47
rc_206.1	4	117.85	0.01	0.00	1.00	117.85	0.01	0.00
rc_206.2	37	862.27[a]	97.01+3185.43	37.69	0.00	862.27	0.00	0.00
rc_206.3	25	714.04	2.46	24.83	17.00	574.42	24.27	19.55
rc_206.4	38	932.69	14.34	35.78	12.00	831.67	63.22	10.83
rc_207.1	34	890.02	42.78	23.35	29.00	732.68	585.65	17.68
rc_207.2	31	901.56	49.87	14.19	24.00	701.25	707.96	22.22
rc_207.3	33	926.36	53.62	14.58	29.00	682.40	492.04	26.34
rc_207.4	6	119.64	0.01	0.00	1.00	119.64	0.38	0.00
rc_208.1	38	931.66	19.90	0.14	21.00	793.61	204.90	14.82
rc_208.2	29	714.55	1.72	0.00	24.00	533.78	1715.26	25.30
rc_208.3	36	828.41	3.41	0.00	37.00	641.31	499.43	22.59
MEDIA		731.24	136.01	27.21	15.48	627.46	351.87	13.55

[a] valore trovato con CPLEX
[b] nessuna soluzione ammissibile trovata

4.1.8 Istanze proposte da Pesant

27 istanze simmetriche proposte da Pesant [Pes98] derivate anch'esse dalle istanze VRPTW (Vehicle Routing Problem with Time Windows) presentate da Solomon [Sol87], ma diverse dalle istanze Potvin e Bengio. Anche in questo caso come per le Potvin - Bengio viene aggiunta la colonna *nodi* che indica il numero di nodi del problema. Inoltre come per la classe precedente abbiamo una matrice (rc208.0) per la quale non otteniamo nessun dato in quanto nè il Simulated Annealing, nè CPLEX riescono a trovare una soluzione feasible. Non avendo valori relativi all'esecuzione di SA+PS questa matrice viene esclusa dal calcolo delle medie per la classe. Il miglioramento della soluzione dovuto al Proximity search è in media del 21.75%, il più alto fra le otto classi prese in considerazione. Per quanto riguarda i tempi di esecuzione, il Proximity Search impiega in media un tempo (622.37) che è circa il triplo di quello del Simulated Annealing (213.98).

Tabella 4.9: Istanze proposte da Pesant

		SA			PS			
nome	nodi	value	time	%elim.	iter	value	time	%migl.
rc201.0	25	752.02	18.36	63.54	9.00	628.62	5.33	16.41
rc201.1	28	673.77[a]	56.33+0.67	65.15	3.00	654.70	7.88	2.83
rc201.2	28	798.12	8.64	65.64	11.00	707.65	14.29	11.34
rc201.3	19	537.63	0.02	58.42	11.00	422.54	6.66	21.41
rc202.0	25	805.93	2.43	27.08	3.00	496.22	34.04	38.43
rc202.1	22	622.04	0.03	34.78	20.00	426.53	31.64	31.43
rc202.2	27	722.21	0.76	42.86	12.00	611.77	44.79	15.29
rc202.3	26	854.14	6.91	33.05	32.00	627.85	32.73	26.49
rc203.0	35	939.35	2.96	11.90	30.00	727.71	415.49	22.53
rc203.1	37	1285.51	98.89	12.52	1.00	819.35	2564.84	36.26
rc203.2	28	890.13	17.48	27.71	19.00	617.47	42.94	30.63
rc204.0	32	872.03	3.74	3.88	60.00	541.45	744.12	37.91
rc204.1	28	777.69	2.76	0.99	43.00	485.37	142.30	37.59
rc204.2	40	944.93	3.96	4.88	45.00	779.17	2603.66	17.54
rc205.0	26	787.92	1.63	38.60	22.00	511.65	13.36	35.06
rc205.1	22	634.78	0.30	47.04	14.00	491.22	10.73	22.62
rc205.2	28	806.52	0.18	47.54	17.00	714.70	69.76	11.38
rc205.3	24	624.70	0.40	43.67	8.00	601.24	25.75	3.76
rc206.0	35	903.92[a]	96.54+3395.73	40.95	0.00	903.92	0	0.00
rc206.1	33	700.68[a]	84.86+705.87	34.05	7.00	664.73	4.38	5.13
rc206.2	32	768.87	56.20	31.72	26.00	655.37	117.28	14.76
rc207.0	37	914.40	14.62	20.41	22.00	806.69	69.99	11.78
rc207.1	33	835.51[a]	80.25+886.19	19.16	16.00	726.36	41.32	13.06
rc207.2	30	757.04	0.50	14.41	18.00	546.41	24.46	27.82
rc208.0	44	—[b]	—	—	—	—	—	—
rc208.1	27	827.46	2.51	36.00	36.00	509.04	2664.65	38.48
rc208.2	29	781.38	12.94	0.00	33.00	503.92	44.31	35.51
MEDIA		800.72	213.98	31.77	19.92	622.37	376.03	21.75

[a] valore trovato con CPLEX

[b] nessuna soluzione ammissibile trovata

4.2 Comparazione delle prestazioni

Ora viene proposta la comparazione fra il metodo di risoluzione descritto in questa tesi che chiamiamo Simulated Annealing - Proximity Search (SAPS) ed altri algoritmi descritti in letteratura per il Problema del Commesso Viaggiatore con Finestre Temporali. Gli euristici considerati per il confronto sono:

- GVNS (General Variable Neighborhood Search) [Mlad13] : algoritmo di ricerca locale basato su scambi 1-shift e 2-opt fra i nodi che costituisce un perfezionamento dell'algoritmo presentato da Da Silva - Urrutia [DaS10], il più recente (2013) ed efficace fra quelli esposti in questa tesi. Questo algoritmo è l'unico per il quale sono disponibili i dati sperimentali per tutte le 8 classi di problemi. Per le istanze delle classi Dumas, Gendreau, Ohlmann-Thomas e Da Silva-Urrutia l'algoritmo è stato applicato 30 volte, mentre per le altre viene applicato 15 volte. Per questo verrà riportato il valore minimo, il valore medio, la deviazione standard dalla media oltre al tempo medio e alla rispettiva deviazione standard.

- TBF (Time Bucket Formulation)[Dash10]: questo algoritmo parte da una diversa formulazione del problema rispetto al classico modello TSPTW introducendo una divisione delle finestre temporali in buckets. L'algoritmo è stato testato per le classi Ascheuer e Pesant. In particolare per la classe Ascheuer vengono riportati solo i valori delle istanze difficili, cioè quelle che richiedono il tempo CPU maggiore. L'algoritmo prevede un tempo limite di cinque ore, pari a 18000 secondi.

- CAH (Compressed Annealing Heuristic)[Ohlm07] : variante del Simulated Annealing chiamata Compressed Annealing, che oltre alla Temperatura introduce il concetto di Pressione. L'algoritmo viene applicato 10 volte per istanza, quindi verranno riportati il valore minimo, il valore medio, la deviazione standard dalla media, il tempo medio di esecuzione e la deviazione standard associata.

- B&C (Branch & Cut)[Asch01] : metodo di risoluzione per istanze asimmetriche basato sul Branch & Cut con una serie di classi di tagli specifiche per il problema considerato, testato sull'unica classe di istanze asimmetriche (Ascheuer). Il time limit è fissato in cinque ore, ed essendo un metodo di risoluzione esatto la soluzione ottenuta è ottima se l'esecuzione termina prima del raggiungimento del time limit, altrimenti viene indicato l'Upper Bound finale come costo della soluzione ottenuta dall'algoritmo.

- GIH (Generalized Insertion Heuristic)[Gen98] : algoritmo euristico basato sull'inserimento dei nodi come metodo di costruzione della soluzione. Partendo dal nodo radice vengono aggiunti ad uno ad uno gli altri nodi applicando una ottimizzazzione locale del percorso costruito. L'algoritmo è stato testato per le istanze delle classi Dumas, Gendreau, Langevin e Potvin-Bengio

- NHC (New Heuristic Calvo)[Cal00] : euristico basato sulla soluzione di un problema ausiliario per ottenere una soluzione vicina all'ammissibilità composta da un tour principale e da tanti piccoli subtour, che poi verranno uniti in un unico tour utilizzando un metodo di greedy inserction. Infine la soluzione viene migliorata utilizzando un algoritmo di ricerca locale. Questo algoritmo viene testato per le classi Dumas, Gendreau, Langevin e Potvin - Bengio.

Per facilitare il confronto fra l'algoritmo presentato in questa tesi ed i diversi algoritmi presi in considerazione, nelle tabelle presentate in seguito le soluzioni migliori di quelle trovate dal Simulated Annealing + Proximity Search sono indicate dal colore blu, mentre quelle peggiori dal rosso. Vengono riportati anche i tempi di esecuzione, anche se non sono molto indicativi viste le differenze fra le macchine utilizzate per i test. Tuttavia come vedremo fra i vari algoritmi si possono notare delle ampie differenze non giustificabili dalle differenti prestazioni dei pc utilizzati, e quindi dovute agli algoritmi testati.

4.2.1 Confronto prestazioni per istanze Dumas

Per queste istanze è disponibile il valore ottimo calcolato da Dumas [Dum95]. Il confronto è fatto con gli algoritmi GVNS, GIH, CAH e NHC. Per quanto riguarda i valori relativi al Generalized Inserction Heuristic [Gen98], non sono disponibili i valori relativi alle istanze n100w40, n100w60 e tutte le istanze con più di 100 nodi. Da un confronto con le soluzioni ottime ricavate da Dumas[Dum95] i valori migliori trovati dal GVNS corrispondono all'ottimo per tutte le istanze. Dai dati vediamo che il nostro metodo per i problemi difficili si comporta generalmente meglio degli altri algoritmi presi in considerazione, escluso il GVNS. In particolare il 50% dei valori relativi al Generalized Insertion Heuristic è peggiore rispetto al SAPS, mentre per quanto riguarda l'euristico proposto da Calvo questa percentuale si abbassa al 40%. Anche i valori medi relativi alle soluzioni ottenute con il Compressed Annealing sono generalmente peggiori per le istanze più difficili, a parte in 3 casi dove sono leggermente migliori. Il General Variable Neighborhood Search ha prestazioni leggermente migliori del SAPS, avendo una media delle soluzioni migliore in 5 casi. Per quanto riguarda i tempi, i valori sono generalmente vicini a quelli degli algoritmi GIH e NHC, mentre sono sempre peggiori rispetto agli algoritmi GVNS e CAH, in particolare per le istanze più difficili.

Tabella 4.10: Confronto prestazioni classe Dumas

	SAPS		GVNS					CAH					GIH		NHC	
nome	value	time	min_value	av_value	st_dev	av_time	st_dev	min_value	av_value	st_dev	av_time	st_dev	value	time	value	time
n20w20	361.20	1.83	361.20	361.20	0.00	0.00	0.00	361.20	361.20	0.00	2.00	0.00	361.20	1.70	361.20	0.00
n20w40	316.00	1.87	316.00	316.00	0.00	0.00	0.00	316.00	316.00	0.00	2.70	0.40	316.00	2.10	316.00	0.00
n20w60	309.80	2.36	309.80	309.80	0.00	0.00	0.00	309.80	309.80	0.00	2.50	0.30	310.20	2.34	309.80	0.00
n20w80	311.00	1.94	311.00	311.00	0.00	0.00	0.00	311.00	311.00	0.00	3.00	0.00	314.20	3.15	311.00	0.00
n20w100	275.20	3.46	275.20	275.20	0.00	0.00	0.00	275.20	275.20	0.00	3.20	0.30	275.20	3.38	275.20	0.00
n40w20	486.60	4.54	486.60	486.60	0.00	0.00	0.00	486.60	486.60	0.00	3.80	0.40	486.60	12.52	486.6	3.00
n40w40	461.00	7.09	461.00	461.00	0.00	0.00	0.00	461.00	461.00	0.00	5.10	0.50	461.00	14.16	461.00	3.00
n40w60	416.40	17.14	416.40	416.40	0.00	0.00	0.00	416.40	416.40	0.00	6.00	0.50	417.00	16.39	416.4	4.80
n40w80	399.80	14.64	399.80	399.80	0.00	1.20	0.60	399.80	399.80	0.00	6.20	0.20	399.80	13.59	399.80	5.20
n40w100	377.00	83.75	377.00	377.00	0.00	0.00	0.00	377.00	377.50	1.20	6.60	0.40	378.80	14.21	377.00	5.60
n60w20	581.60	8.65	581.60	581.60	0.00	0.00	0.00	581.60	581.60	0.00	7.20	0.90	581.60	36.54	581.60	8.40
n60w40	590.20	19.67	590.20	590.20	1.90	0.10	0.00	590.20	590.70	2.00	8.20	0.40	590.20	36.79	590.40	17.20
n60w60	560.00	29.53	560.00	560.00	0.00	0.00	0.00	560.00	560.00	0.20	8.50	0.40	567.00	44.29	560.00	20.20
n60w80	508.00	88.68	508.00	508.00	0.00	0.20	0.20	508.00	509.30	1.60	8.60	0.30	517.20	32.93	509.0	18.00
n60w100	514.80	899.42	514.80	514.80	0.00	0.10	0.10	514.80	516.50	3.00	8.60	0.40	524.00	49.75	516.4	26.20
n80w20	676.60	109.03	676.60	676.60	0.00	0.00	0.00	676.60	676.60	0.00	11.30	0.40	676.60	74.03	676.60	43.40
n80w40	630.00	23.26	630.00	630.00	0.00	0.10	0.00	630.00	630.20	0.60	11.50	0.40	630.40	66.32	630.40	69.20
n80w60	607.60	88.82	606.40	606.70	1.20	1.00	0.60	606.40	607.00	1.40	12.00	0.30	616.80	89.13	—[a]	—
n80w80	595.40	411.27	593.80	593.90	0.20	0.60	0.80	593.80	594.10	2.30	11.50	0.50	601.40	75.75	594.40	59.60
n100w20	757.60	26.71	757.60	757.60	0.00	0.00	0.00	757.60	757.80	0.80	15.40	0.40	757.60	174.98	757.80	102.60
n100w40	701.80	69.16	701.80	701.80	0.00	0.10	0.00	701.80	702.40	1.10	15.70	0.30	—[b]	—	703.60	128.60
n100w60	698.80	439.34	696.60	696.60	0.00	0.10	0.10	696.60	697.20	1.20	15.90	0.50	—[b]	—	696.60	148.00
n150w20	868.40	331.45	868.40	868.40	0.00	0.10	0.10	868.40	869.20	1.30	24.70	0.70	—[b]	—	868.60	419.80
n150w40	834.80	310.02	834.80	834.80	0.00	0.40	0.40	834.80	836.20	2.70	25.20	0.70	—[b]	—	837.40	529.60
n150w60	820.00	1080.21	818.60	818.60	0.00	1.90	0.80	818.80	820.20	4.80	25.60	0.70	—[b]	—	820.40	630.00
n200w20	1009.80	1151.81	1009.00	1009.10	0.20	2.00	1.20	1009.00	1010.00	2.10	35.10	1.70	—[b]	—	1010.00	1456.20
n200w40	984.20	2907.41	984.20	984.20	0.21	5.20	1.40	984.60	986.10	3.40	35.20	1.30	—[b]	—	985.40	2105.80

[a] media non presente in quanto per una delle cinque matrici della sottoclasse non viene trovata nessuna soluzione ammissibile
[b] valori non disponibili

4.2.2 Confronto prestazioni per istanze Gendreau

Come per le Dumas, il confronto è fatto con gli algoritmi GVNS, GIH, CAH e NHC. Analizzando le medie delle soluzioni ottenute per gli algoritmi GVNS e CAH, per il GVNS sono sempre migliori tranne per i problemi più semplici dove sono uguali ai nostri risultati (SAPS), mentre per il CAH in 5 casi i nostri risultati sono migliori ma per altre 17 serie di istanze sono peggiori. Anche per quanto riguarda gli ultimi 2 algoritmi presi in considerazione per il confronto (GIH e NHC), le soluzioni con costo peggiore di quelle trovate dal SAPS sono in numero superiore rispetto a quelle di costo migliore (4 peggiori e 24 migliori per il GIH, 10 peggiori e 18 migliori per il NHC). I tempi del SAPS per questa classe non sono comparabili a quelli degli altri algoritmi analizzati, in quanto nettamente più alti.

Tabella 4.11: Confronto prestazioni classe Gendreau

	SAPS		GVNS					CAH					GIH		NHC	
nome	valueSAPS	timeSAPS	min_value	av_ value	st_dev	av_time	st_dev	min_value	av_value	st_dev	av_time	st_dev	value	time	value	time
n20w120	265.60	9.94	265.60	265.60	0.00	0.00	0.00	265.60	265.60	0.00	3.10	0.40	269.20	4.08	267.20	0.00
n20w140	232.80	7.47	232.80	232.80	0.00	0.00	0.00	232.80	232.80	0.00	3.90	0.30	263.80	4.37	259.60	0.00
n20w160	218.20	6.03	218.20	218.20	0.00	0.00	0.00	218.20	218.20	0.00	4.00	0.10	261.20	4.77	260.00	0.00
n20w180	236.60	110.66	236.60	236.60	0.00	0.00	0.00	236.60	236.60	0.00	4.00	0.10	259.80	5.98	244.60	0.00
n20w200	241.00	231.59	241.00	241.00	0.00	0.00	0.00	241.00	241.00	0.00	4.10	0.20	245.20	6.31	243.00	0.40
n40w120	377.80	78.30	377.80	377.80	0.00	0.00	0.00	377.80	378.10	1.10	6.00	0.20	372.80	18.38	360.80	4.80
n40w140	364.40	715.15	364.40	364.40	0.00	0.00	0.00	364.40	364.70	1.60	6.00	0.10	356.20	18.86	348.40	9.40
n40w160	326.80	949.51	326.80	326.80	0.00	0.00	0.00	326.80	327.10	0.60	6.00	0.20	348.00	20.00	337.20	10.20
n40w180	334.40	672.31	330.40	330.50	0.90	2.20	1.20	332.00	333.90	2.30	6.20	0.40	328.20	17.02	326.80	12.40
n40w200	314.40	371.93	313.80	313.80	0.30	3.60	1.20	313.80	315.00	1.00	6.30	0.40	326.20	522.80	315.20	16.20
n60w120	451.00	992.86	451.00	451.00	0.00	0.30	0.20	451.00	452.90	2.80	8.30	0.20	492.00	51.59	483.40	29.80
n60w140	453.20	1136.78	452.00	452.00	0.00	0.10	0.00	452.40	454.00	2.10	8.60	0.40	454.80	49.47	454.40	28.00
n60w160	466.00	88.11	464.00	464.60	0.20	0.00	0.00	464.60	465.40	2.30	8.40	0.40	451.60	47.53	448.60	33.80
n60w180	428.20	379.38	421.20	421.20	0.00	0.40	0.20	421.60	425.20	4.40	8.60	0.40	439.20	52.26	432.80	40.60
n60w200	430.20	504.96	427.40	427.40	0.00	0.30	0.10	427.40	430.80	5.00	8.40	0.30	439.60	43.50	428.00	57.00
n80w100	580.00	1201.75	578.60	578.60	0.00	0.70	0.40	579.20	581.60	2.40	11.50	0.40	584.20	99.56	580.20	72.80
n80w120	544.00	587.58	541.40	541.40	0.10	1.30	0.80	541.40	544.00	2.10	11.50	0.40	581.80	121.02	549.80	64.00
n80w140	507.40	570.55	506.00	506.30	0.60	1.40	0.50	509.80	513.60	4.70	11.30	0.40	555.20	94.17	525.60	75.20
n80w160	506.20	1309.46	504.80	505.10	1.20	1.50	1.10	505.40	511.70	5.20	11.20	0.40	524.80	85.69	502.80	82.20
n80w180	503.20	2621.69	500.60	500.90	2.30	3.30	1.00	502.00	505.90	4.00	11.40	0.30	511.00	99.01	489.00	116.20
n80w200	485.20	1747.13	481.80	481.80	0.00	0.40	0.20	481.80	486.40	4.00	11.10	0.30	508.60	112.34	484.00	158.20
n100w80	666.40	364.98	666.40	666.40	0.00	0.40	0.20	666.40	668.10	2.60	15.90	0.40	675.60	118.14	668.00	139.20
n100w100	643.40	742.85	640.60	641.00	1.50	2.80	1.10	642.20	645.00	2.60	14.60	0.50	671.20	129.54	644.00	118.60
n100w120	600.40	890.81	597.20	597.50	0.50	5.60	1.70	601.20	603.70	2.10	15.00	0.40	624.60	204.17	614.40	167.40
n100w140	548.80	698.78	548.40	548.40	0.00	0.20	0.00	579.20	582.50	3.20	14.90	0.40	634.60	207.67	591.40	200.60
n100w160	556.40	2438.09	555.00	555.00	0.00	1.10	0.30	584.00	588.80	3.80	15.00	0.60	585.20	215.57	570.40	214.20
n100w180	570.20	2394.32	561.60	561.60	0.00	1.20	0.60	561.60	566.90	4.60	14.90	0.40	585.20	225.12	566.00	244.60
n100w200	566.40	2426.37	550.20	550.60	3.97	4.00	1.00	555.40	562.30	5.80	14.90	0.40	588.60	168.24	555.60	242.00

4.2.3 Confronto prestazioni per istanze Ohlmann - Thomas

Il confronto delle prestazioni viene fatto con gli algoritmi GVNS e CAH. Per questa classe di istanze sia i tempi di esecuzione dell'algoritmo SAPS, sia i risultati ottenuti sono peggiori rispetto al GVNS e al CAH, a parte per due medie sulle soluzioni relative al CAH. In particolare la differenza è del 2% circa per i dati relativi alle soluzioni delle due serie di problemi con 200 nodi, il divario maggiore fra le classi composte da serie di matrici divise per numero di nodi e media delle finestre temporali (Dumas, Gendreau, Ohlmann - Thomas, Da Silva - Urrutia, Langevin).

Tabella 4.12: Confronto prestazioni classe Ohlmann - Thomas

	SAPS		GVNS					CAH				
nome	value	time	min_value	av_value	st_dev	av_time	st_dev	min_value	av_value	st_dev	av_time	st_dev
n150w120	728.80	2428.65	722.00	722.10	0.60	11.20	3.60	725.00	731.10	5.50	24.80	0.90
n150w140	711.00	3214.44	693.80	693.90	0.40	18.70	4.40	697.60	705.40	6.70	24.90	0.70
n150w160	680.40	3227.39	671.00	672.60	2.90	13.40	5.00	673.60	680.90	5.90	25.00	1.00
n200w120	821.80	3349.11	803.60	803.90	0.30	11.50	3.60	806.80	817.00	7.00	34.40	1.30
n200w140	824.40	3624.01	798.00	798.70	1.60	24.70	5.90	804.60	812.60	6.70	35.20	1.00

4.2.4 Confronto prestazioni per istanze Da Silva - Urrutia

Per questa classe di istanze, introdotta nel 2012, vengono riportati solamente i dati relativi all'applicazione del GVNS. Per queste istanze sarebbero disponibili anche i dati relativi al VNS descritto proprio dai due autori che hanno introdotto per la prima volta queste istanze [DaS10]. Tuttavia i due algoritmi hanno lo stesso principio di funzionamento, cambiano solamente alcuni dettagli implementativi che rendono il GVNS più performante, per questo viene considerato soltanto quest'ultimo per il confronto. Osservando i risultati ottenuti dal SAPS e dal GVNS, mentre i tempi di esecuzione sono nettamente migliori per il GVNS, i valori trovati (considerando la media sui 30 run effettuati per matrice dal GVNS) possono spingere ad equiparare i due algoritmi. Infatti il GVNS per 10 volte trova un valore migliore rispetto al SAPS e per 9 volte peggiore. Tuttavia per i problemi più difficili, cioè con numero di nodi e dimensione delle finestre temporali elevati, il divario si allarga a favore del GVNS con differenze superiori all'1% del valore delle soluzioni trovate dal SAPS.

Tabella 4.13: Confronto prestazioni classe Da Silva - Urrutia

	SAPS		GVNS				
nome	value	time	min_value	av_ value	st_dev	av_time	st_dev
n200w100	10019.60	67.29	10019.60	10020.40	0.80	0.00	0.00
n200w200	9252.00	57.52	9252.00	9254.20	11.40	0.10	0.00
n200w300	8022.80	648.83	8022.80	8023.10	0.30	10.00	3.30
n200w400	7079.20	2468.30	7062.40	7072.40	19.30	11.80	3.70
n200w500	6484.40	1166.39	6466.20	6472.70	11.40	13.80	4.20
n250w100	12633.00	64.79	12633.00	12633.00	0.00	0.00	0.00
n250w200	11310.40	79.47	11310.40	11314.00	5.00	0.30	0.10
n250w300	10230.40	305.13	10230.40	10231.00	3.40	3.70	1.90
n250w400	8900.40	1620.29	8896.20	8897.90	5.30	37.70	7.70
n250w500	8123.40	2393.85	8069.80	8083.50	13.20	42.20	8.10
n300w100	15041.20	181.99	15041.20	15041.20	0.00	0.00	0.00
n300w200	13851.40	124.12	13851.40	13857.60	14.90	0.60	0.20
n300w300	11479.40	1047.75	11477.20	11478.80	2.70	10.90	3.40
n300w400	10427.60	2368.28	10402.80	10419.60	25.50	30.00	6.00
n300w500	10001.60	3166.47	9842.20	9849.20	7.90	49.50	6.30
n350w100	17494.00	246.53	17494.00	17494.00	0.00	0.00	0.00
n350w200	15672.00	618.71	15672.00	15672.00	0.00	1.70	0.90
n350w300	13648.60	1083.34	13648.80	13660.80	17.80	13.20	3.80
n350w400	12157.80	2383.44	12083.20	12090.60	9.50	46.80	7.90
n350w500	11484.00	3109.83	11347.80	11360.60	17.70	59.00	7.50
n400w100	19454.80	268.83	19454.80	19454.80	0.00	0.00	0.00
n400w200	18439.80	147.87	18439.80	18442.60	5.10	1.80	0.40
n400w300	15871.80	684.84	15871.80	15875.80	8.50	28.80	4.80
n400w400	14110.80	3311.09	14079.40	14112.00	24.40	54.90	6.90
n400w500	12850.40	3697.24	12716.60	12755.80	26.90	77.50	7.80

4.2.5 Confronto prestazioni per istanze Langevin

Il confronto è fatto con gli algoritmi GVNS, GIH, CAH e NHC. Per quanto riguarda la serie n60ft20 non abbiamo a disposizione il valore ottenuto dal Generalized Insertion Heuristc in quanto per una delle matrici della serie l'algoritmo non riesce a trovare una soluzione ammissibile. Per questa classe di istanze che possiamo considerare facile date le prestazioni degli algoritmi considerati, non si registrano significative differenze nei valori delle soluzioni trovate. Questo fa pensare che, tranne per qualche istanza fra quelle con 60 nodi, tutti gli algoritmi arrivino all'ottimo. Per quanto riguarda i tempi il SAPS è il peggiore, anche se resta comunque sotto i 5 minuti per tutte le serie di istanze.

Tabella 4.14: Confronto prestazioni classe Langevin

	SAPS		GVNS					CAH					GIH		NHC	
nome	valueSAPS	timeSAPS	min_value	av_ value	st_dev	av_time	st_dev	min_value	av_value	st_dev	av_time	st_dev	value	time	value	time
n20ft30	724.70	9.79	724.70	724.70	0.00	0.00	0.00	724.70	724.70	0.00	2.40	0.80	724.70	1.73	724.70	0.00
n20ft40	721.50	13.63	721.50	721.50	0.00	0.00	0.00	721.50	721.50	0.00	3.40	0.60	721.50	1.71	721.50	0.00
n40ft20	982.70	81.04	982.70	982.70	0.00	0.00	0.00	982.70	982.70	0.00	4.40	1.20	982.70	11.55	982.70	0.25
n40ft40	951.80	111.04	951.80	951.80	0.00	0.00	0.00	951.80	951.80	0.00	4.70	1.60	951.80	13.93	951.80	0.60
n60ft20	1215.70	264.18	1215.70	1215.70	0.00	0.00	0.00	1215.70	1215.70	0.00	5.60	2.40	—[a]	—	1215.70	5.00
n60ft30	1183.20	265.60	1183.20	1183.20	0.00	0.00	0.00	1183.20	1183.20	0.30	8.10	2.70	1183.50	28.23	1183.20	5.00
n60ft40	1160.80	267.54	1160.70	1160.70	0.00	0.00	0.00	1160.80	1160.80	0.80	9.00	2.10	1160.80	30.63	1160.80	10.90

[a] media non presente in quanto per una delle dieci matrici della sottoclasse non viene trovata nessuna soluzione ammissibile

4.2.6 Confronto prestazioni per istanze Ascheuer

Per questa classe sono disponibili i risultati relativi agli algoritmi GVNS, TBF e B&C. Questi problemi nella loro formulazione originale presentano dei tempi di servizio dei nodi diversi da zero. Nella nostra formulazione e in quella GVNS si è passati al modello senza tempi di servizio sommandoli al costo degli archi in uscita dal nodo al quale fanno riferimento. Negli altri due algoritmi presi in considerazione non si è fatta questa modifica, quindi il costo finale delle soluzioni è più basso in quanto il valore degli archi è minore, dato che il loro costo non comprende i tempi di servizio. Per rendere confrontabili i costi delle soluzioni ottenute è stata aggiunta la somma dei tempi di servizio dei nodi ai valori delle soluzioni ottenute dagli algoritmi TBF e B&C. La correttezza di questa somma deriva dal fatto che la soluzione del TSP con finestre temporali è un tour che tocca una sola volta tutti i nodi del grafo, quindi ad ogni nodo corrisponde un solo arco in uscita. Nella formulazione utilizzata in questa tesi ogni arco aumenterà la soluzione del valore pari al tempo di servizio del nodo di uscita, quindi il costo finale aumenterà di una quantità pari alla somma dei tempi di servizio di tutti i nodi del grafo.

Per molte istanze non sono disponibili i dati relativi al TBF. Questo avviene per due motivi opposti: o il problema rientra fra quelli definiti facili, e per scelta degli autori vengono omessi i relativi risultati, oppure è un problema difficile per il quale non viene trovata alcuna soluzione ammissibile. Esaminando i dati relativi alle soluzioni trovate (av_value per il GVNS) vediamo che per le istanze considerate 'easy' da Dash [Dum95] non ci sono differenze fra i valori ottenuti dai vari algoritmi tranne per l'applicazione del B&C alla $rbg125a$. Per quanto riguarda le istanze rimanenti, l'algoritmo migliore è il GVNS, mentre il B&C ha prestazioni significativamente peggiori del SAPS sia per valori che per tempi di esecuzione. Il TBF per 6 volte trova una soluzione migliore rispetto a quella del SAPS, ma per 4 volte non trova alcuna soluzione entro il time limit di 5 ore. Anche in questo caso il GVNS risulta essere l'algoritmo più veloce, mentre il B&C è il più lento arrivando per 18 volte al time limit.

Tabella 4.15: Confronto prestazioni classe Ascheuer

	SAPS		GVNS					TBF		B&C	
nome	value	time	min_value	av_ value	st_dev	av_time	st_dev	value	time	value	time
rbg010a	671.00	1.91	671.00	671.00	0.00	0.00	0.00	—[a]	—	671.00	0.12
rbg016a	938.00	0.92	938.00	938.00	0.00	0.00	0.00	—[a]	—	938.00	0.20
rbg016b	1304.00	6.18	1304.00	1304.00	0.00	0.00	0.00	—[a]	—	1304.00	8.80
rbg017	893.00	1.56	893.00	893.00	0.00	0.00	0.00	—[a]	—	893.00	0.82
rbg017.2	852.00	3.92	852.00	852.00	0.00	0.00	0.00	—[a]	—	852.00	0.30
rbg017a	4296.00	2.78	4296.00	4296.00	0.00	0.00	0.00	—[a]	—	4296.00	0.12
rbg019a	1262.00	1.30	1262.00	1262.00	0.00	0.00	0.00	—[a]	—	1262.00	0.03
rbg019b	1866.00	7.60	1866.00	1866.00	0.00	0.00	0.00	—[a]	—	1866.00	54.57
rbg019c	4536.00	23.27	4536.00	4536.00	0.00	0.00	0.00	—[a]	—	4536.00	8.72
rbg019d	1356.00	1.36	1356.00	1356.00	0.00	0.00	0.00	—[a]	—	1356.00	0.75
rbg020a	4689.00	2.90	4689.00	4689.00	0.00	0.00	0.00	—[a]	—	4689.00	0.20
rbg021	4536.00	23.79	4536.00	4536.00	0.00	0.00	0.00	—[a]	—	4536.00	8.75
rbg021.2	4528.00	14.31	4528.00	4528.00	0.00	0.00	0.00	—[a]	—	4528.00	0.22
rbg021.3	4528.00	53.75	4528.00	4528.00	0.00	0.00	0.00	—[a]	—	4528.00	27.15
rbg021.4	4525.00	23.71	4525.00	4525.00	0.00	0.00	0.00	—[a]	—	4525.00	5.82
rbg021.5	4515.00	8.17	4515.00	4515.00	0.00	0.00	0.00	—[a]	—	4515.00	6.63
rbg021.6	4480.00	5.99	4480.00	4480.00	0.00	0.00	0.00	—[a]	—	4480.00	1.38
rbg021.7	4479.00	13.94	4479.00	4479.00	0.00	0.00	0.00	—[a]	—	4479.00	4.30
rbg021.8	4478.00	6.32	4478.00	4478.00	0.00	0.00	0.00	—[a]	—	4478.00	17.40
rbg021.9	4478.00	16.47	4478.00	4478.00	0.00	0.00	0.00	—[a]	—	4478.00	26.12
rbg027a	5091.00	20.57	5091.00	5091.00	0.00	0.00	0.00	—[a]	—	5091.00	2.25
rbg031a	1863.00	15.43	1863.00	1863.00	0.00	0.00	0.00	—[a]	—	1863.00	1.70
rbg033a	2069.00	15.61	2069.00	2069.00	0.00	0.00	0.00	—[a]	—	2069.00	1.85
rbg034a	2222.00	16.70	2222.00	2222.00	0.00	0.00	0.00	—[a]	—	2222.00	0.98
rbg035a	2114.00	18.85	2144.00	2144.00	0.00	0.00	0.00	—[a]	—	2144.00	1.83
rbg035a.2	2056.00	117.04	2056	2056.00	0.00	0.00	0.00	—[a]	—	2056.00	64.80
rbg038a	2480.00	31.55	2480.00	2480.00	0.00	0.00	0.00	—[a]	—	2480.00	4232.23
rbg040a	2378.00	2164.61	2378.00	2378.00	0.00	0.00	0.00	—[a]	—	2378.00	751.82
rbg041a	2598.00	187.09	2598.00	2598.00	0.00	9.00	11.00	2598.00	146.80	2613.00	18000.00
rbg042a	2774.00	583.38	2772.00	2772.93	0.04	14.00	21.00	2772.00	188.30	2796.00	18000.00
rbg048a	9428.00	673.09	9383.00	9383.00	0.00	1.00	1.00	9383.00	129.20	9423.00	18000.00
rbg049a	10050.00	2036.19	10018.00	10018.50	0.01	6.00	15.00	10020.00	18000.00	10035.00	18000.00
rbg050a	2953.00	1690.48	2953.00	2953.27	0.03	11.00	17.00	—[a]	—	2953.00	18.62
rbg050b	9889.00	1938.56	9863.00	9863.00	0.00	6.00	8.00	—[b]	18000.00	9893.00	18000.00
rbg050c	10033.00	768.08	10024.00	10024.00	0.00	2.00	3.00	—[b]	18000.00	10034.00	18000.00
rbg055a	3761.00	37.35	3761.00	3761.00	0.00	0.00	0.00	—[a]	—	3761.00	6.40
rbg067a	4625.00	69.84	4625.00	4625.00	0.00	0.00	0.00	—[a]	—	4625.00	5.95
rbg086a	8400.00	220.46	8400.00	8400.00	0.00	1.00	1.00	8400.00	4.90	8401.00	18000.00
rbg092a	7158.00	1414.55	7158.00	7158.00	0.00	5.00	6.00	7158.00	90.40	7176.00	18000.00
rbg125a	7936.00	209.66	7936.00	7936.00	0.00	0.00	0.00	—[a]	—	7937.00	229.82
rbg132	8468.00	511.52	8468.00	8468.00	0.00	6.00	6.00	8468.00	37.60	8508.00	18000.00
rbg132.2	8211.00	1737.18	8191.00	8191.73	0.01	11.00	17.00	8191.00	2761.10	8233.00	18000.00
rbg152	10032.00	351.89	10032.00	10032.00	0.00	11.00	16.00	10032.00	43.70	10041.00	18000.00
rbg152.3	9788.00	1888.77	9788.00	9788.60	0.01	16.00	18.00	9788.00	10353.30	9843.00	18000.00
rbg172a	10950.00	2567.35	10950.00	10951.20	0.01	19.00	20.00	10950.00	425.50	11048.00	18000.00
rbg193	12535.00	3305.52	12535.00	12540.20	0.03	20.00	17.00	12535.00	159.60	12573.00	18000.00
rbg193.2	12164.00	3467.03	12138.00	12141.10	0.02	19.00	15.00	—[b]	18000.00	12214.00	18000.00
rbg201a	12994.00	2545.33	12948.00	12950.10	0.02	23.00	16.00	12948.00	462.70	13055.00	18000.00
rbg233	15002.00	2528.58	14993.00	15001.30	0.03	33.00	17.00	14992.00	749.40	15089.00	18000.00
rbg233.2	14591.00	3619.89	14494.00	14497.60	0.03	33.00	13.00	—[b]	18000.00	14607.00	18000.00

[a] istanza semplice per la quale non vengono forniti i risultati computazionali
[b] nessuna soluzione ammissibile trovata entro il time limit

4.2.7 Confronto prestazioni per istanze Potvin - Bengio

Il confronto dei risultati riguardanti le istanze proposte da Potvin e Bengio viene fatto con tre degli algoritmi presi in considerazione: il General Variable Neighbourhood Search (GVNS), il Generalized Insertion Heuristic (GIH) proposto da Gendreau e l'euristico proposto da Calvo (NHC). Non considerando il problema *rc*_204.1 per il quale SAPS a differenza degli altri algoritmi presi in esame non trova alcuna soluzione, GVNS migliora i risultati ottenuti dal SAPS in sei casi, mentre peggiora in un caso. Per gli altri due algoritmi (GIH e NHC) il rapporto si inverte: le soluzioni migliori rispetto al SAPS sono due per entrambi mentre quelle peggiori sono rispettivamente tredici ed otto. Inoltre il NHC non trova alcun valore per due istanze. I tre algoritmi considerati per il confronto sono da uno a due ordini di grandezza più veloci rispetto al SAPS; comunque tranne che per tre casi l'esecuzione del SAPS non va oltre i quindici minuti.

Tabella 4.16: Confronto prestazioni classe Potvin - Bengio

		SAPS		GVNS					GIH		NHC	
nome	n_nodi	value	time	min_value	av_ value	st_dev	av_time	st_dev	value	time	value	time
rc_201.1	20	444.54	5.46	444.54	444.54	0.00	0.00	0.00	444.54	3.00	444.54	0.00
rc_201.2	26	711.54	5.00	711.54	711.54	0.00	0.00	0.00	712.91	6.98	711.54	0.00
rc_201.3	32	790.61	77.03	790.61	790.61	0.00	0.00	0.00	795.44	14.98	790.61	3.00
rc_201.4	26	793.64	45.12	793.64	793.64	0.00	0.00	0.00	793.64	6.98	793.64	0.00
rc_202.1	33	771.78	2845.20	771.78	771.78	0.00	14.00	12.00	772.18	10.55	722.18	8.00
rc_202.2	14	304.14	12.05	304.14	304.14	0.00	0.00	0.00	304.14	2.35	304.14	0.00
rc_202.3	29	839.58	23.55	837.72	837.72	0.00	0.00	0.00	839.58	6.97	839.58	0.00
rc_202.4	28	793.03	52.90	793.03	793.03	0.00	0.00	0.00	793.03	11.55	793.03	2.00
rc_203.1	19	453.48	29.44	453.48	453.48	0.00	0.00	0.00	453.48	4.03	453.48	0.00
rc_203.2	33	784.16	61.78	784.16	784.16	0.00	0.00	0.00	784.16	15.67	784.16	4.00
rc_203.3	37	821.21	358.53	817.53	817.53	0.00	0.00	0.00	842.25	16.02	819.42	14.00
rc_203.4	15	314.29	5.07	314.29	314.29	0.00	0.00	0.00	314.29	2.98	314.29	0.00
rc_204.1	46	—[a]	—	878.64	878.64	0.00	0.00	0.00	897.09	26.43	868.76	35.00
rc_204.2	33	662.16	288.86	662.16	662.16	0.00	0.00	0.00	679.26	15.90	665.96	8.00
rc_204.3	24	460.47	689.22	455.03	455.03	0.00	0.00	0.00	460.24	11.18	555.03	4.00
rc_205.1	14	343.21	7.09	343.21	343.21	0.00	0.00	0.00	343.21	1.13	343.21	0.00
rc_205.2	27	755.93	76.22	755.93	755.93	0.00	0.00	0.00	755.93	7.33	755.93	0.00
rc_205.3	35	825.06	98.33	825.06	825.06	0.00	0.00	0.00	825.06	42.90	825.06	21.00
rc_205.4	28	760.47	1624.92	760.47	760.47	0.00	0.00	0.00	762.41	6.58	—[b]	—
rc_206.1	4	117.85	0.02	117.85	117.85	0.00	0.00	0.00	117.85	0.01	117.85	0.00
rc_206.2	37	862.27	0.00	828.06	828.06	0.00	0.00	0.00	842.17	33.47	853.31	10.00
rc_206.3	25	574.42	26.73	574.42	574.42	0.00	0.00	0.00	591.20	6.75	574.42	0.00
rc_206.4	38	831.67	77.56	831.67	832.06	0.18	1.00	4.00	845.04	31.48	837.54	8.00
rc_207.1	34	732.68	628.43	732.68	732.68	0.00	0.00	0.00	741.53	14.76	733.22	4.00
rc_207.2	31	701.25	757.83	701.25	701.25	0.00	0.00	0.00	718.09	16.28	—[b]	—
rc_207.3	33	682.40	545.66	682.40	682.40	0.00	0.00	0.00	684.40	17.25	687.28	10.00
rc_207.4	6	119.64	0.39	119.64	119.64	0.00	0.00	0.00	119.64	0.01	119.64	0.00
rc_208.1	38	793.61	224.80	789.25	791.86	0.28	1.00	2.00	799.19	26.58	789.25	10.00
rc_208.2	29	533.78	1716.98	533.78	533.78	0.00	0.00	0.00	543.41	20.53	537.33	2.00
rc_208.3	36	641.31	502.84	634.44	634.44	0.00	0.00	0.00	660.15	25.63	649.11	8.00

[a] valore non disponibile per la mancanza di una soluzione di partenza per PS ammissibile
[b] nessuna soluzione trovata

4.2.8 Confronto prestazioni per istanze Pesant

Per queste istanze sono disponibili i dati relativi a due degli algoritmi presi in considerazione: il General Variable Neighbourhood Search ed il Time Bucket Formulation. Come per la classe Ascheuer, anche in questo caso ci sono delle differenze nei costi legate al fatto di sommare o meno il tempo di servizio dei nodi al costo dei lati in uscita. Infatti per rendere i valori relativi all'applicazione del TBF comparabili con gli altri dati, verrà sommato al costo delle soluzioni il tempo totale di servizio dei nodi. Per questa classe di istanze non notiamo grandi differenze fra le soluzioni restituite dagli algoritmi persi in esame. Il GVNS per le istanze rc203.2 ed rc204.2 migliora la soluzione del SAPS di qualche decimale. Per quanto riguarda altre due istanze (rc203.1 ed rc206.0) sia il GVNS sia il TBF giungono a soluzioni migliori di più del 5% rispetto a quelle relative al SAPS. Infine l'algoritmo TBF non ottiene alcuna soluzione ammissibile per i problemi rc204.2 e rc208.0. Per quanto riguarda i tempi di esecuzione, il GVNS non va oltre qualche secondo mentre il TBF nella maggior parte dei casi risulta più veloce del SAPS. Comunque notiamo che tranne per cinque problemi il SAPS non supera i quindici minuti.

Tabella 4.17: Confronto prestazioni classe Pesant

		SAPS		GVNS					TBF	
nome	n_nodi	value	time	min_value	av_ value	st_dev	av_time	st_dev	value	time
Rc201.0	25	628.62	23.69	628.62	628.62	0.00	0.00	0.00	628.62	0.00
Rc201.1	28	654.70	64.88	654.70	654.70	0.00	0.00	0.00	654.70	0.00
Rc201.2	28	707.65	22.93	707.65	707.65	0.00	0.00	0.00	707.65	0.00
Rc201.3	19	422.54	6.68	422.54	422.54	0.00	0.00	0.00	422.54	0.10
Rc202.0	25	496.22	36.47	496.22	496.22	0.00	0.00	0.00	496.22	0.20
Rc202.1	22	426.53	31.67	426.53	426.53	0.00	0.00	0.00	426.53	0.10
Rc202.2	27	611.77	45.55	611.77	611.77	0.00	0.00	0.00	611.77	0.10
Rc202.3	26	627.85	39.64	627.85	627.85	0.00	0.00	0.00	627.85	0.20
Rc203.0	35	727.71	418.45	727.45	727.45	0.00	0.00	0.00	727.45	3437.70
Rc203.1	37	819.35	2663.73	726.99	726.99	0.00	0.00	0.00	726.99	2722.70
Rc203.2	28	617.47	60.42	617.46	617.46	0.00	0.00	0.00	617.47	0.90
Rc204.0	32	541.45	747.86	541.45	541.45	0.00	0.00	0.00	541.45	2.80
Rc204.1	28	485.37	145.06	485.37	485.37	0.00	0.00	0.00	485.37	14.90
Rc204.2	40	779.17	2607.62	778.40	778.40	0.00	2.00	5.00	—[b]	18000.00
Rc205.0	26	511.65	14.99	511.65	511.65	0.00	0.00	0.00	511.65	0.10
Rc205.1	22	491.22	11.03	491.22	491.22	0.00	0.00	0.00	491.22	0.10
Rc205.2	28	714.70	69.94	714.70	714.70	0.00	0.00	0.00	714.70	0.30
Rc205.3	24	601.24	26.15	601.24	601.24	0.00	0.00	0.00	601.24	0.10
Rc206.0	35	904.43	3492.27	835.23	835.23	0.00	3.00	3.00	835.23	73.30
Rc206.1	33	664.73	795.11	664.73	664.73	0.00	0.00	0.00	664.73	85.00
Rc206.2	32	655.37	173.48	655.37	655.37	0.00	0.00	0.00	655.37	84.80
Rc207.0	37	806.69	84.61	806.69	806.69	0.00	0.00	0.00	806.69	19.00
Rc207.1	33	726.36	1007.76	726.36	726.36	0.00	0.00	0.00	726.36	34.90
Rc207.2	30	546.41	24.96	546.41	546.41	0.00	0.00	0.00	546.41	116.40
Rc208.0	44	—[a]	—	820.56	820.56	0.00	1.00	4.00	—[b]	18000.00
Rc208.1	27	509.04	2667.16	509.04	509.04	0.00	0.00	0.00	509.04	990.60
Rc208.2	29	503.92	57.25	503.92	503.92	0.00	0.00	0.00	503.92	7.90

[a] valore non disponibile per la mancanza di una soluzione di partenza per PS ammissibile
[b] nessuna soluzione trovata entro il time limit

4.3 Variante Simulated Annealing

Dalle prove effettuate osserviamo che il Proximity Search in genere porta ad un miglioramento significativo della soluzione ottenuta dal Simulated Annealing, in particolare per i problemi più difficili. Questo miglioramento può dipendere, oltre che dall'efficacia dell'algoritmo, anche dalla scarsa qualità della soluzione restituita dal Simulated Annealing. Quindi per valutare l'efficacia del Proximity Search l'idea è quella di testarlo con delle soluzioni di partenza migliori di quelle ottenute con l'esecuzione del Simulated Annealing. In questa sezione testiamo il Proximity Search descritto nel capitolo 2 sulle istanze della classe Da Silva - Urrutia partendo dalle soluzioni trovate con il Variable Neighbourhood Search proposto dagli stessi autori [DaS10]. Le soluzioni sono tutte disponibili nel sito $http://homepages.dcc.ufmg.br/rfsilva/tsptw/$, anche se in due casi le soluzioni fornite sono risultate essere non ammissibili. Queste soluzioni si avvicinano alle migliori soluzioni attualmente disponibili per questo tipo di istanze, cioè quelle ottenute con il GVNS [Mlad13]. Dai dati ottenuti vediamo che per 14 delle 25 sottoclassi tramite il PS otteniamo dei miglioramenti alle soluzioni di partenza. Il miglioramento massimo si verifica per la sottoclasse di problemi più difficili, la n400w500, ed è del 0.17% con una media di 4.8 iterazioni per istanza. La media sulla classe è di 1.16 iterazioni per un miglioramento dello 0.03% ottenuto in 202.29 secondi. Confrontando questi dati con quelli relativi al SAPS, Proximity Search in questa variante è più veloce, esegue meno iterazioni ed il miglioramento rispetto alla soluzione iniziale è minore, come era logico attendersi data la nuova soluzione iniziale a costo significativamente inferiore della precedente ottenuta con il Simulated Annealing.

Tabella 4.18: Variante PS classe Da Silva- Urrutia

	GVS		PS			
nome	value	time	value	time	iter	%migl.
n200w100	10019.60	4.80	10019.60	0.04	1.00	0.00
n200w200	9252.00	5.80	9252.00	4.91	1.00	0.00
n200w300	8026.40	7.20	8022.80	33.25	0.80	0.04
n200w400	7067.20	8.70	7062.40	399.76	1.00	0.07
n200w500	6466.40	10.00	6466.20	0.00	0.00	0.00
n250w100	12633.00	9.90	12633.00	0.24	1.00	0.00
n250w200	11310.40	11.90	11310.40	9.10	0.80	0.00
n250w300	10230.40	14.90	10230.40	0.00	0.00	0.00
n250w400	8899.20	18.90	8896.20	824.46	1.00	0.03
n250w500	8082.40	20.70	8082.40	0.00	0.00	0.00
n300w100	15041.20	21.20	15041.20	0.08	1.00	0.00
n300w200[a]	13900.75	23.70	13898.50	30.86	1.00	0.02
n300w300[b]	11475.25	37.00	11472.25	28.03	0.50	0.03
n300w400	10413.00	31.70	10402.80	573.65	3.00	0.10
n300w500	9861.80	35.40	9851.00	556.18	2.60	0.11
n350w100	17494.00	41.00	17494.00	0.17	1.00	0.00
n350w200	15672.00	47.30	15672.00	7.85	0.40	0.00
n350w300	13650.20	54.90	13648.80	79.03	1.20	0.01
n350w400	12099.00	60.20	12088.20	304.30	1.60	0.09
n350w500	11365.80	57.80	11365.00	109.38	0.40	0.01
n400w100	19454.80	57.10	19454.80	0.13	1.00	0.00
n400w200	18439.80	66.90	18439.80	0.00	0.00	0.00
n400w300	15873.40	93.60	15871.80	79.44	1.20	0.01
n400w400	14115.40	96.20	14096.20	639.43	2.80	0.14
n400w500	12747.60	109.30	12726.00	1376.89	4.80	0.17
MEDIA	12143.64	37.84	12139.91	202.29	1.16	0.03

[a] la matrice n300w200.005 è esclusa dalla media in quanto la soluzione di partenza è errata

[b] la matrice n300w300.001 è esclusa dalla media in quanto la soluzione di partenza è errata

Conclusioni

L'algoritmo proposto in questa tesi si rivela performante sotto l'aspetto del costo delle soluzioni trovate, questo grazie alla procedura Proximity Search che permette di migliorare velocemente una soluzione anche se molto distante dall'ottimo. Il lato negativo è che il Simulated Annealing + Proximity Search per quel che riguarda i tempi di esecuzione non regge il confronto con i più recenti algoritmi euristici sviluppati in modo specifico per il Problema del Commesso Viaggiatore con Finestre Temporali come il General Variable Neighbourhood Search. Tuttavia a differenza della gran parte degli euristici presi in esame in questa tesi, il SA + PS è un algoritmo applicabile ad un qualsiasi tipo di problema modellabile come MIP. Infatti adattando il concetto di vicinanza di una soluzione al problema preso in esame, dopo aver fatto un corretto tuning dei parametri sia il Simulated Annealing che il Proximity Search sono facilmente applicabili ad un qualsiasi problema del quale sia disponibile un modello a variabili intere. Quindi l'algoritmo presentato in questa tesi potrebbe essere utilizzato come metodo efficace di ricerca di soluzioni per problemi MIP poco studiati per i quali è computazionalmente difficile la soluzione all'ottimo e non esiste un euristico efficiente. Per quanto riguarda la variante al PS proposta alla fine del capitolo 4, i dati ottenuti dimostrano che Proximity Search è utile anche per migliorare delle soluzioni già di per sè vicine all'ottimo, quindi data la sua generalità può costituire un valido postprocessing da applicare alla soluzione ottenuta da un qualsiasi metodo di risoluzione che non assicuri l'ottimalità della soluzione trovata.

Bibliografia

[DaS10] R. Ferreira da Silva and S. Urrutia. A General VNS heuristic for the traveling salesman problem with time windows, Discrete Optimization, v.7, Issue 4, pp.203-211, 2010

[Dash10] S. Dash, O. Günlük, A. Lodi, and A. Tramontani. A Time Bucket Formulation for the Traveling Salesman Problem with Time Windows, INFORMS Journal on Computing, v.24, pp.132-147, 2012 (published online before print on December 29, 2010)

[Pot96] Y. Potvin, S. Bengio. The Vehicle Routing Problem with Time Windows Part II: Genetic Search. INFORMS Journal on Computing, v.8, pp.165-172, 1996

[Sol87] Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows. Operations Research, v.35, pp.254-265, 1987

[Lang93] Langevin et al. A Two-Commodity Flow Formulation for the Traveling Salesman and Makespan Problems with Time Windows, Networks, v.23, pp.631-640, 1993

[Dum95] Dumas et al. An Optimal Algorithm for the Traveling Salesman Problem with Time Windows, Operations Research, v.43, pp.367-371, 1995

[Gen98] M. Gendreau, A. Hertz, G. Laporte, M. Stan. A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows. Operations Research, v.43, pp.330-335, 1998

[Ohlm07] J.W. Ohlmann, B.W. Thomas, A compressed-annealing heuristic for the traveling salesman problem with time windows. INFORMS Journal on Computing, v.19, pp.80-90, 2007

[Asch01] N. Ascheuer, M. Fischetti, M. Grötschel, Solving asymmetric travelling salesman problem with time windows by branch-and-cut. Mathematical Programming, v.90, pp.475-506, 2001

[Mlad13] N. Mladenovic, R. Todosijevic, D. Urosevic, An efficient general variable neighborhood search for large traveling salesman problem with time windows. Yugoslav Journal of Operations Research, v.23, pp.19-30, 2013

[Cal00] R. Calvo, A New Heuristic for the Traveling Salesman Problem with Time Windows. Transportation Science, v.34, i.1, pp.113-124, 2000

[Pes98] G. Pesant, M. Gendreau, J.Y. Potvin, J.M. Rousseau. An exact constraint logic programming algorithm for the travelling salesman problem with time windows. Transportation Science, v.32, pp.12-29, 1998

[Kirk84] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies. Journal of Statistical Physics, v.34, i.5-6, pp.975-986, 1984

[Fis13] M. Fischetti, M. Monaci, Proximity Search for 0-1 Mixed-Integer Convex Programming, 31 July 2013