UNIVERSITÀ DEGLI STUDI DI MILANO

Facoltà di Scienze Matematiche, Fisiche, Naturali

CORSO DI LAUREA IN INFORMATICA

Confronto sperimentale tra diversi intorni in un algoritmo di ricerca tabu per il problema del commesso viaggiatore con operazioni miste di consegna e raccolta

RELATORE
Prof.Giovanni Righini

TESI DI LAUREA DI
Emanuele Tresoldi
Matricola: 618193

Anno Accademico 2003/2004

Prefazione

La tesi proposta tratta il confronto sperimentale tra algoritmi tabu search per l'approssimazione del problema del commesso viaggiatore con operazioni miste di consegna e raccolta (TSPDC), variante del noto problema del commesso viaggiatore.

Inizialmente è descritto il problema, ne viene data una definizione matematica e viene proposta una panoramica degli algoritmi esistenti per la risoluzione del TSPDC.

Nella seconda parte vengono descritti gli algoritmi utilizzati per il confronto, i principi sui quali si basano e la loro implementazione sia che si tratti di algoritmi originali sia che siano algoritmi già esistenti.

Infine nell'ultima parte è presentata una catalogazione ed analisi dei risultati sperimentali ottenuti. Vengono confrontati i vari algoritmi sia per quanto concerne la qualità delle soluzioni ottenute sia per i tempi d'esecuzione.

Ringrazio il Professor Giovanni Righini,
per l'aiuto, la pazienza e la cordialità
dimostratemi durante lo sviluppo
della tesi.

CONFRONTO SPERIMENTALE TRA DIVERSI INTORNI IN UN ALGORITMO DI RICERCA TABU PER IL PROBLEMA DEL COMMESSO VIAGGIATORE CON OPERAZIONI MISTE DI CONSEGNA E RACCOLTA

1. Introduzione

1.1. Il problema del TSP with distribution and collection

Il travelling salesman problem (TSP)[1] è un noto problema di ottimizzazione NP-Hard nel quale, dato un grafo $G=(V, E)$ in cui l'insieme dei vertici $V=\{1, \ldots, N\}$ rappresenta le città e una matrice NxN, associata all'insieme degli archi $E = \{(i,j): i \neq j;\ i,j \in V\}$, nella quale l'elemento $[i,j]$ indica la distanza dalla città i alla città j; si richiede di trovare il ciclo hamiltoniano (ciclo che tocca tutti i nodi del grafo una ed una sola volta) di costo minimo. Il TSP, date le sue numerose applicazioni pratiche, è un problema molto studiato e sono state considerate numerose varianti (con capacità, con finestre temporali, con grafo asimmetrico ed altre).

Il problema del commesso viaggiatore con operazioni miste di consegna e raccolta (traveling salesman problem with distribution and collection o TSPDC) è definito nel modo seguente: sia $G=(V_0, A)$ un grafo completo e non orientato. $V_0 = \{0\} \cup V$ è l'insieme dei clienti (con $V=\{1,\ldots,N\}$).
Il vertice 0 rappresenta il deposito mentre gli altri vertici $i \in V$ rappresentano i clienti da servire. Ogni utente richiede una operazione di consegna di una quantità $d_i \geq 0$ di merce proveniente dal magazzino e una operazione di carico di merce $p_i \geq 0$ da riportare al magazzino. Ad ogni arco $(i, j) \in A$, $i, j \in V_0$ è associato un costo non negativo c_{ij}. Il TSPDC consiste nel determinare un ciclo hamiltoniano che inizi e termini al deposito, nel quale staziona un veicolo di capacità Q, che serva esattamente una volta ogni cliente, che non ecceda la capacità del veicolo e che minimizzi la lunghezza, definita come la somma del costo degli archi che compongono il ciclo.
Le operazioni di consegna e carico della merce, allo stesso cliente, non devono avvenire separatamente e l'operazione di consegna è da considerarsi eseguita per prima.

Per garantire la risolvibilità del problema si assume che $P = \sum_{i=1}^{n} p_i \leq Q$ e $D = \sum_{i=1}^{n} d_i \leq Q$. Si assume inoltre che il grafo G sia completo e che la matrice NxN dei costi sia triangolarizzata, più precisamente

$$c_{ik} + c_{kj} \geq c_{ij} \text{ per ogni } i, j, k \in V_0$$

Questa proprietà garantisce che la ottima soluzione TSPDC sia un ciclo Hamiltoniano che visita ogni cliente una sola volta.
Il TSP può essere considerato una generalizzazione del TSPDC infatti se si pone $p_i = d_i$ per ogni $i \in V$ il TSPDC si riduce al TSP.

Questo problema ha numerose applicazioni pratiche nella progettazione e gestione dei sistemi di distribuzione, a questo proposito si veda ad esempio Mosheiov[2].

1.2. Proprietà delle soluzioni TSPDC ammissibili

Per ogni cliente $i \in V$, è definita la quantità $\delta_i = p_i - d_i$ che rappresenta la richiesta del cliente *i*. Si osservi che δ_i può essere negativa, ciò significa che la quantità di merce da consegnare al cliente *i* è maggiore della quantità di merce da prelevare dal medesimo cliente. Esistono dunque due tipi di clienti: quelli con δ_i negativo che svuotano il veicolo e quelli con δ_i non negativo che riempiono il veicolo.
Si consideri un sottoinsieme $S \subseteq V$ di clienti visitati consecutivamente e sia $\Delta_s = \sum_{i \in S} \delta_i$ il costo addizionale (positivo o negativo) associato alla visita, in qualsiasi ordine, di tutti i clienti appartenenti ad *S*. Sia, inoltre, *L* il carico totale del veicolo, in una soluzione ammissibile, prima di visitare *S*. Se l'ordine di visita dei clienti, appartenenti ad *S*, non è specificato valgono le seguenti proprietà.

Proprietà 1. In ogni soluzione ammissibile $0 \le L \le Q - \max\{0, \Delta_s\}$.

Dimostrazione. Il veicolo lascia il deposito con carico D, quindi lungo qualsiasi circuito si ha $L \ge 0$. Se $\Delta_s > 0$ allora ci deve essere nel veicolo abbastanza spazio per inserire il carico addizionale associato ad *S*, altrimenti, indipendentemente dall'ordine nel quale S è visitato il veicolo termina la visita dei clienti in *S* violando il vincolo di capacità *Q*. Se $\Delta_s \le 0$ allora qualsiasi $L \le Q$ può portare ad una soluzione ammissibile poiché il veicolo terminerà la visita di *S* senza mai eccedere la capacità *Q*.

Proprietà 2. Preso $S \subseteq V$ ed L tali che la Proprietà 1 sia valida, allora, esiste sempre un cammino ammissibile che visiti tutti i clienti appartenenti ad S e che rispetti il vincolo di capacità.

Dimostrazione. Si consideri il cammino che visita tutti i clienti con δ_i negativo prima di visitare quelli con δ_i non negativo. Assumendo che $\sum_{i \in S} p_i \le Q$ e $\sum_{i \in S} d_i \le Q$,questo cammino è ammissibile poiché $L \le Q - \max\{0, \Delta_S\}$

Si assuma ora che i clienti in *S* siano visitati nell' ordine $\{i_1, i_2, \ldots \ , i_{|S|}\}$. Allora è valida la seguente proprietà.

Proprietà 3. In ogni soluzione ammissibile si ha $L \le Q - \max\left\{0, \max\left\{\sum_{j=1}^{h} \delta_i, h = 1, \ldots, |S|\right\}\right\}$.

2. Algoritmi esistenti per il TSPDC

Il TSPDC, come ogni variante del TSP, è un problema piuttosto studiato. Non sono, però, molti gli algoritmi disponibili per la sua risoluzione.

Mosheiov[2] ha proposto un modello matematico per il TSPDC e degli algoritmi euristici basati sull'estensione dei metodi già usati per il TSP. È stato dimostrato che per uno degli algoritmi proposti l'errore nel caso peggiore è uguale a $1+\alpha$, dove α è l'errore nel caso peggiore della metaeuristica, per il TSP, sul quale si basa. Anily e Mosheiov[3] hanno descritto un nuovo algoritmo con errore nel caso peggiore pari a 2 basato sulla soluzione di alberi ricoprenti minimi.

Un caso speciale di TSPDC, conosciuto come Traveling Salesman Problem with Backhauls (TSPB), dove ogni cliente richiede solo o una operazione di consegna o una operazione di carico e dove tutti i clienti che richiedono operazioni di consegna sono visitati prima di quelli con operazioni di carico è stata studiata da Gendreau, Hertz e Laporte[4], che presentano una estensione per il TSPB della metaeuristica GENIUS per il TSP[5].

La generalizzazione del TSPDC correlata al Vehicl Routing Problem (VRP), nella quale molti veicoli sono disponibili, è stata studiata da Halse[6] che ha proposto una formulazione matematica ed un algoritmo euristico basato sul rilassamento lagrangeano del problema.

Tzoreff, Granot, Granot e Sosic hanno ottenuto alcuni risultati per il TSPDC su grafi con struttura particolare[7].

Un tabu search per l'approssimazione del TSPDC, che è il punto di partenza della mia tesi, è stato sviluppato da Gendreau, Laporte e Vigo [8].

Recentemente Baldacci, Hadjicostantinou e Mingozzi hanno sviluppato un algoritmo per la risoluzione ottimale del TSPDC [9].

3. Metodi di generazione dell'intorno

Gli algoritmi utilizzati per la mia tesi appartengono alla tipologia della ricerca locale[10] ai quali è applicata una metaeuristica, la tabu search nel mio caso, allo scopo di ottimizzarne il funzionamento e i risultati ottenuti. Vengono ora presentate le tecniche di ricerca locale utilizzate nella tesi, mentre, la tabu search è spiegata più avanti.

Gli algoritmi di ricerca locale presi in esame nella mia tesi si basano sugli intorni; questi algoritmi vengono inizializzati con una soluzione iniziale *S.* Ad ogni iterazione, partendo da *S*, esplorano tutte le soluzioni appartenenti al intorno *N(S)* e ci si sposta sulla soluzione migliore che diviene la nuova S.

L'intorno *N(S)* può essere ottenuto in molti modi, in particolare io ho analizzato due tecniche: 2-scambi e 3-scambi. È stato preso in considerazione, inoltre, un terzo metodo ad intorno variabile ottenuto dalla combinazione dei primi due.

3.1 2-scambi

Questa tecnica genera un intorno contenente tutte le soluzioni ammissibili ottenute dalla sostituzione di due archi con altri due. Viene generata una soluzione per ogni coppia di archi e quindi il numero delle soluzioni esaminate sarà pari a $O(N^2)$.

Presa una soluzione ammissibile S (Figura 3.1), un 2-scambio associato agli archi (a, a_1) e (b, b_1) genera la soluzione visibile in Figura 3.2 contenente gli archi (a, b) e (a_1, b_1) e nella quale l'orientamento di tutti gli archi compresi tra *b* e a_1 è cambiato. La nuova soluzione sarà ammissibile se la sequenza invertita tra *b* e a_1 è ammissibile. In particolare, detto L_a il carico del veicolo dopo aver visitato *a,* la nuova sequenza è ammissibile se $L_a + \sum_{j=b}^{h} \delta_j \leq Q$ per ogni *h* appartenente alla sequenza compresa tra *b* e a_1.

Il costo computazionale per l'esplorazione completa di questo intorno è $O(N^2)$.

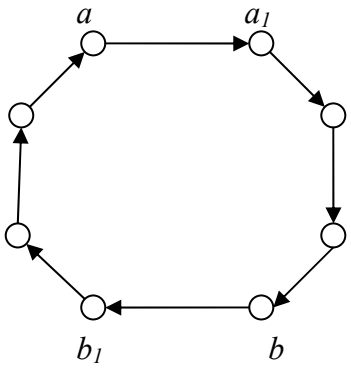


Figura 3.1

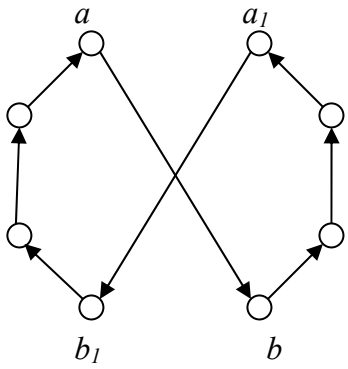


Figura 3.2

3.2 3-scambi

La tecnica dei 3-scambi è simile, per principio, a quella dei 2-scambi, ma, a differenza di quest'ultima, prevede che siano generate tutte le soluzioni date dalla sostituzione di tre archi con altri tre. L'intorno cosi generato avrà dimensione $O(N^3)$.

Un esempio di 3-scambio associato alla rottura degli archi (a,a_1), (b,b_1) e (c,c_1) è visibile nella Figura 3.3

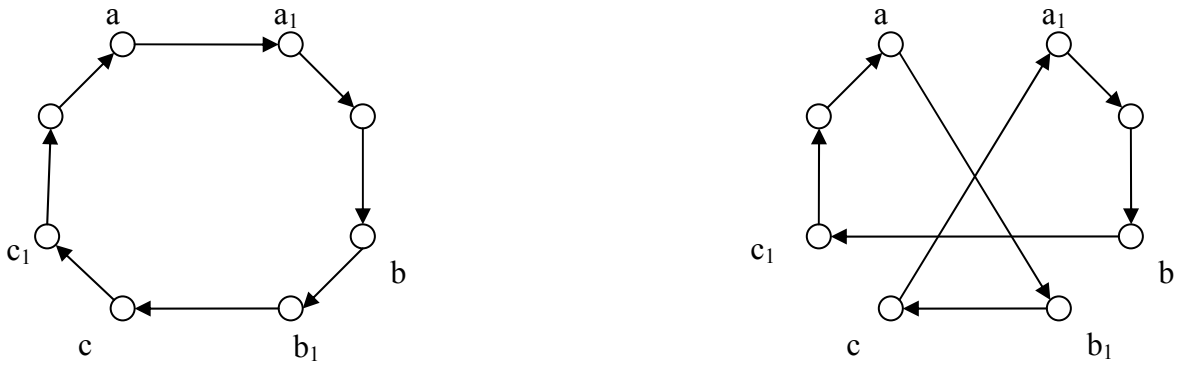


Figura 3.3

In questo caso, al contrario del 2-scambi, nessun arco è cambiato di orientamento. Ciò elimina il problema dell'ammissibilità delle sequenze invertite infatti l'ammissibilità di questa soluzione è dovuta solo ai nuovi archi inseriti.

Il costo per l'esplorazione completa dell'intorno così generato è $O(N^3)$.

4. Tabu search

La tabu search[11] è una metaeuristica di ricerca locale. Queste tecniche nascono per ovviare al problema degli ottimi locali. Nel momento in cui un algoritmo di ricerca locale classico giunge in un ottimo locale, si ferma. Occorre invece proseguire la ricerca altrove, nel tentativo di trovare altri ottimi locali, e possibilmente un ottimo globale.

Per superare questo problema la tabu search introduce memoria nell'algoritmo, in particolare memorizza alcune informazioni sulle ultime soluzioni visitate, orientando la ricerca in modo da cercare di uscire dagli ottimi locali. È necessario, quindi, accettare mosse che possono portate ad un peggioramento della soluzione corrente, pur di spostarsi sufficientemente lontano dal minimo locale. Se l'algoritmo si trova in un minimo locale x, dal quale non è possibile migliorare ulteriormente, allora la tabu search prevede che ci si sposti sulla soluzione y appartenente all'intorno di x per la quale è minimo il peggioramento della funzione obbiettivo. A questo punto è possibile che la migliore soluzione disponibile nell'intorno di y sia x, e cioè quella soluzione dalla quale ci si vuole allontanare.

La tabu search per evitare il problema del ciclaggio memorizza, in una tabu list, le ultime mosse che hanno portato alla soluzione attuale etichettandole come tabu. Si impedisce all'algoritmo di effettuare per un certo periodo, chiamato *tabu tenure*, una mossa che è etichettata come tabu al fine di evitare ricadute in ottimi locali già visitati.

La tabu list è gestita come una coda; infatti ad ogni iterazione la mossa che era nella tabù list da maggior tempo viene cancellata e torna ad essere considerata valida.

La tabu search è combinabile con qualunque algoritmo di ricerca locale, nel mio caso essa è stata abbinata al 2-scambi ed al 3-scambi. Il questo ambito, la metaeuristica tabu search, guida la direzione della ricerca locale impedendo di effettuare scambi tra archi che sono stati inseriti recentemente nella soluzione.

5. Implementazione

In questo capitolo, dopo la spiegazione dei metodi di costruzione del ciclo iniziale, verrà descritto come le tecniche di ricerca locale e la tabu search sono state implementate in tre differenti algoritmi.

Il primo è una semplice riscrittura della tabu search di Gendreau, Laporte e Vigo[8].

Il secondo ed il terzo, invece, sono stati sviluppati da me.

5.1 Costruzione del ciclo iniziale

Ogni algoritmo di ricerca locale necessita di una soluzione iniziale. Su di essa infatti, viene generato l'intorno che sarà poi visitato alla ricerca di una soluzione migliore.

Il primo metodo analizzato è semplicissimo e consiste nell'ordinare tutte le città da visitare in ordine di δ_i. È facile vedere come il ciclo iniziale così prodotto sia ammissibile: a partire dal deposito vengono, infatti, visitate prima tutte le città con δ_i negativo per poi passare a quelle con δ_i non negativo. Il vincolo di carico, quindi, non potrà mai essere violato.

È stato inoltre analizzato un secondo metodo; dopo aver generato un ciclo iniziale ordinando le città in base al δ_i come in precedenza divide il ciclo in due sottosequenze distinte. La prima S_1 formata dalla città con δ_i negativo e la seconda S_2 formata dalle città con δ_i non negativo. Ogni sequenza viene ordinata nuovamente nel modo seguente. La prima città di S_1 è il deposito, la seconda città, quella appartenente ad S_1 con distanza dal deposito minore, la terza quella con distanza minore dalla seconda città e così via per tutte le città di S_1. S_2 è costruita in modo analogo ad S_1 con la sola differenza che la prima città è quella con distanza minore dall'ultima inserita in S_1. Infine S è data dalla concatenazione di S_1 e S_2. Anche in questo caso il vincolo di carico non potrà mai essere violato e si ottiene una soluzione iniziale con costo minore.

Non ho ritenuto di dover utilizzare algoritmi complessi per ottenere dei cicli iniziali migliori poiché lo scopo della tesi è quello di confrontare più algoritmi e quindi, data una medesima inizializzazione - seppur migliorabile - per tutti gli algoritmi, il confronto non perde di validità.

5.2 L'algoritmo tabu search di Gendreau, Laporte e Vigo

Gendreau, Laporte e Vigo hanno sviluppato nel 1998 un algoritmo tabu search per il TSPDC che fa uso di intorni generati tramite 2-scambi[8].

L'algoritmo, per implementare il meccanismo tabu search, ad ogni iterazione memorizza i nuovi archi inseriti nella soluzione. Ogni movimento che rimuove questi archi è considerato tabu per le successive iterazioni. La lista tabu è stata realizzata da Gendreau, Laporte e Vigo mediante una matrice *N*x*N* che, per ogni arco, memorizza la più recente iterazione nella quale è stato incluso nella soluzione. Questo valore è impostato a $-\infty$ ad ogni riavvio.

La tabu tenure *t* è inizialmente scelta casualmente nell'intervallo $[t_{\min}, t_{\max}]$.

$t_{\min} = \max\{2, N/4\}$ mentre $t_{\max} = 5N/4$. Ogni 25 iterazioni, il valore di *t* è aggiornato; in particolare se durante le ultime 25 iterazioni la soluzione ottima è stata migliorata allora $t = \min\{t+2, t_{\max}\}$, altrimenti $t = \max\{t-2, t_{\min}\}$.

L'algoritmo è diviso in due parti: nella prima, inizializzata tramite una soluzione ammissibile nota, la ricerca termina dopo 200 iterazioni consecutive che non migliorano l'ottimo, la seconda parte è inzializzata con la soluzione della prima e termina dopo 500 iterazioni non miglioranti.

L'implementazione di questo algoritmo fatta da Gendreau Laporte e Vigo è scritta in FORTRAN77.

Io, ho reimplementato l'algoritmo scrivendolo in ANSI C. Nella mia implementazione è stato, inoltre, modificato il parametro t_{max} da *5N/4* a *N/2*. Questa modifica è stata eseguita in seguito a numerose prove sperimentali. Ho osservato, infatti, che mantenendo *t* al valore originale, l'algoritmo svolgeva delle iterazioni nelle quali non era possibile compiere alcuno scambio poiché tutti gli archi erano etichettati come tabu. Si prenda ad esempio una grafo con *N*=12 e conseguentemente t_{max} =15. Se in fase di inizializzazione $t = t_{max}$ dopo sei iterazioni la tabu list sarà piena non permettendo all'algoritmo di compiere scambi fino alla sedicesima iterazione. Per questo motivo ho ritenuto di dover diminuire t_{max} fino a *N/2,* valore per il quale nella tabu table ci saranno sempre almeno due archi liberi.

Questo approccio è in grado di fornire della buone soluzioni in tempi brevi ed è dunque stato considerato da Gendreau, Laporte e Vigo il migliore compromesso tra qualità delle soluzioni e tempo computazionale impiegato.

In figura 5.1 è riportato un semplice schema di funzionamento, in pseudocodice, dell'algoritmo.

```
2-scambi(iterazioni_max,time_out,S)

{

t=rand(t_min,t_max);

  while(i < iterazioni_max && tempo < time_out)
  {
        ∀Z ∈ N(S) - Z ∈ (tabu_list)

              Zp= min{Z ∈ N(S)}


              if (Zp < S_best)
                    {
                          Z* = Zp;
                          S = Zp;
                          i = 0;
                    }
              elseif (Zp < S)
                    {
                          S = Zp ;
                          i ++;
                    }
              elseif (Zp > S )
                    {
                          S = Zp;
                          i ++;
                    }
              iterazioni++;

              aggiorna_tabu_list(Zp);

              if (iterazione%25==0)
                    aggiorna_t();

              for (i=0; i<N;i++)
                    for (j=0;j<N;j++)
                          if (iterazione – tabu_list [i][j]>tabu_tenure)
                                rilascia_arco(a_ij)
  }
        return Z*;
}
```

Figura 5.1 – Pseudocodice 2-scambi

5.3 Tabu search con 3-scambi

Ho realizzato in linguaggio C un algoritmo tabu search simile a quello realizzato da Gendreau, Laporte e Vigo; la differenza risiede nella tecnica utilizzata per la generazione dell'intorno: io ho utilizzato il 3-scambi invece del 2-scambi. Il meccanismo della tabu search non ha subito variazioni se non per quanto riguarda il valore di t_{max} che ho impostato pari a N/3 per i medesimi motivi illustrati nel paragrafo precedente.

Di questo algoritmo sono state realizzate diverse versioni preliminari con differenti metodi di esplorazione dell'intorno. Sono state considerate le seguenti politiche.

- Best improve: viene analizzato completamente l'intorno e si sceglie la migliore soluzione sulla quale spostarsi.
- First improve: si analizza l'intorno finché non si trova la prima soluzione migliorante, e ci si sposta su quella. In caso non sia possibile trovare soluzioni miglioranti, l'intorno viene analizzato completamente e viene fatta la scelta migliore.
- Esplorazione parziale: l'intorno viene esplorato solo per una percentuale determinata a priori e vengono prese in considerazione solo le soluzioni appartenenti a quella porzione di intorno.

Tra tutte queste politiche la best improve offre, come ovvio, soluzioni migliori, seppure con tempi di esecuzione più lunghi rispetto alle altre due, le quali, degradando la qualità delle soluzioni, riescono a diminuire i tempi d'esecuzione. Preferendo la qualità della soluzione al tempo di calcolo nella versione definitiva dell'algoritmo è usata la politica best improve.

È stata, inoltre, considerata l'introduzione di criteri d'aspirazione: questo significa che l'algoritmo può effettuare una mossa tabu se questa porta ad una "buona soluzione". Nel mio caso "buona soluzione" è stato tradotto con soluzione migliore dell'ottimo conosciuto. I criteri d'aspirazione sono stati introdotti per cercare di correggere l'eccessiva restrittività che in alcuni casi è stata riscontrata nella tabu search. Nella versione definitiva dell'algoritmo questi criteri d'aspirazione non sono stati introdotti poiché non portavano a miglioramenti concreti della qualità delle soluzioni.

Il tabu search con 3-scambi è stato sviluppato con lo scopo di ottenere delle soluzioni migliori rispetto a quelle ottenibili utilizzando tabu search con 2-scambi. Esso ha, comunque, il difetto di richiedere un tempo computazionale molto più elevato rispetto al 2-scambi.

5.4 Tabu search con intorno variabile

Sempre in linguaggio C, ho realizzato infine un terzo algoritmo tabu searchche utilizza una combinazione di 2-scambi e 3-scambi per la generazione dell'intorno e permette di realizzare intorni di dimensione variabile; più precisamente di dimensione $O(N^2)$ o $O(N^3)+ O(N^2)$. Il loro costo d'esplorazione, ovviamente, sarà rispettivamente $O(N^2)$ e $O(N^3)+ O(N^2)$.

Partendo da una soluzione iniziale S l'algoritmo analizza l'intorno generato tramite 2-scambi. Se trova una soluzione migliorante vi si sposta e passa alla prossima iterazione; in caso contrario viene generato anche l'intorno prodotto dal 3-scambi. I due intorni vengono analizzati, le soluzioni migliori di entrambi gli intorni vengono confrontate ed è scelta la soluzione migliore tra le due.

Questo algoritmo è stato realizzato per cercare di combinare la velocità del 2-scambi con la maggiore potenza del 3-scambi in modo da ottenere un algoritmo più robusto ed equilibrato che permettesse soluzioni migliori del 2-scambi con tempi inferiori a quelli del 3-scambi.

In Figura 5.2 si può trovare uno schema di funzionamento dell'algoritmo con intorno variabile.

```
Intorno_variabile(iterazioni_max,time_out,S)

{

t=rand(t_min,t_max);

  while(i < iterazioni_max && tempo < time_out)
  {
        ∀Z ∈ N(S) - Z ∈ (tabu_list)

              Zp= min{Z ∈ N(S)}


              if (Zp < S_best)
                    {
                          Z* = Zp;
                          S = Zp;
                          i = 0;
                    }
              elseif (Zp < S)
                    {
                          S = Zp ;
                          i ++;
                    }
              elseif (Zp > S )
                    {
                          Zp3=3-scambi(S)

                          if (Zp3 > Zp)
                          {
                                S = Zp;
                                i ++;
                          }
                          else
                          {
                                if (Zp3 < S_best)
                                {
                                      Z* = Zp3;
                                      S = Zp3;
                                      i = 0;
                                }
                                else
                                {
                                      S = Zp3 ;
                                      i ++;
                                }

                          }
                    }
```

```
            iterazioni++;

            aggiorna_tabu_list(Zp);

            if (iterazione%25==0)
                  aggiorna_t();

            for (i=0; i<N;i++)
                  for (j=0;j<N;j++)
                        if (iterazione – tabu_list [i][j]>tabu_tenure)
                              rilascia_arco(a_ij)
  }
      return Z*;
}
```

Figura 5.2 – Pseudocodice Tabu search con intorno variabile

6. Risultati sperimentali

Gli algoritmi sono stati codificati in ANSI C e sono stati compilati con Dev C++ versione 4.9.9.1. La macchina usata per i test è un AMD Athlon XP 2.09 GHz con 1GB di RAM e sistema operativo Windows XP.

6.1 Files in input

I programmi ricevono in input un file di testo (Figura 6.1).

Nella prima parte del file sono riportati il nome della mappa, il numero di nodi, il tipo di mappa, il metodo di rappresentazione degli archi e le coordinate nel piano di tutte le città.

Nella seconda sono riportati il valore di β, il numero di città (escluso il deposito), il carico massimo Q ed infine i valori di d_i e p_i di tutte le città.

```
NAME : V15D
COMMENT : TSPDC instance generated by GLV (1999)
TYPE : TSP
DIMENSION :   16
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
  1   30   40
  2   37   52
  3   49   49
  4   52   64
  5   20   26
  6   40   30
  7   21   47
  8   17   63
  9   31   62
 10   52   33
 11   51   21
 12   42   41
 13   31   32
 14    5   25
 15   12   42
 16   36   16
EOF

 0.2000000          15         1        258        230  0.1100006
          8         23         19         7         25         11
         22         18         13         3         22         23
         27         16         12
          7         30         16         9         21         15
         19         23         11         5         19         29
         23         21         10
```

Figura 6.1 – Input file

I files sono stati gentilmente concessi dal professor Roberto Baldacci. Sono gli stessi sui quali Baldacci, Hadjicostantinou e Mingozzi hanno realizzato i test contenuti in [9].

Grazie a questi files è stato possibile realizzare dei test che fossero confrontabili direttamente con dei risultati esistenti in modo da verificare la bontà del mio approccio al problema.

I risultati presenti in questa sezione fanno riferimento a tre differenti classi di problemi utilizzate per i test.

La prima classe di problemi (Classe A) è costituita da istanze TSPDC derivate da istanze di problemi di vehicl routing (VRP). Per ogni problema VRP è stata ricavata un'istanza TSPDC nella quale l'insieme dei clienti, il deposito e la matrice dei costi sono le stesse dell'istanza VRP. Le quantità d_i e p_i dell'istanza TSPDC sono generate a partire dalla richiesta r_i associata ad ogni cliente dell'istanza VRP secondo la formula seguente:

$$d_i = r_i \qquad p_i = \begin{cases} \lfloor (1-\beta) r_i \rfloor & \text{se } i \text{ è pari} \\ \lfloor (1+\beta) r_i \rfloor & \text{se } i \text{ è dispari} \end{cases} \qquad i = 1, \ldots, n, \quad \text{(Formula 6.1)}$$

β è un numero non negativo più piccolo di 1. Dunque la δ_i d'ogni cliente i è data da d_i-p_i;, inoltre si ha che $Q = \max\{D, C\}$. Per questa classe di problemi ho considerato 6 differenti VRP con n compreso tra 25 e 199; per ognuna sono state generate le istanze TSPDC corrispondenti a β = 0.05, 0.10 e 0.20.

La Classe B è formata da problemi euclidei dove le coordinate del deposito e di ogni cliente sono generate casualmente nell'intervallo [0,100] e il costo d'ogni arco del grafo G è dato dalla distanza euclidea tra i suoi estremi.

La Classe C è costituita da problemi simmetrici nei quali il costo di ogni arco del grafo G è generato casualmente nell'intervallo [0,100]. In seguito la matrice di costi è derivata applicando l'algoritmo di Floyd-Warshall al grafo G.

Per quanto concerne le classi B e C sono stati considerati problemi con n= 25, 50, 75, 100, 150 e 200. Il valore d_i d'ogni cliente i è stato generato in modo casuale tra [1,100] mentre q_i è generato in funzione della Formula 6.1 con i medesimi valori di β utilizzati per i problemi di Classe A. Sono state, inoltre, generate delle istanze con valori d_i e q_i non correlati tra loro, ma entrambi generati casualmente nell'intervallo [1,100]. Queste istanze sono contraddistinte nelle tabelle da $\beta = \infty$.

Per ogni istanza TSPDC i tre algoritmi sono stati lanciati 20 volte ed è stato imposto un time-out di 3600 secondi sia all'esecuzione del tabu search con 3-scambi sia a quello con intorno variabile.

Nelle tabelle, sotto riportate, sono utilizzati i seguenti valori:

- *Mappa* : indica il nome della mappa
- β : indica il valore di β utilizzato per generare l'istanza in analisi
- Z^* : indica l'ottimo individuato da Baldacci in [9]
- Z : indica la media delle soluzioni ottime da me ottenute
- Z_{locale} : indica la soluzione ottima ottenuta tramite ricerca locale senza tabu search
- σ : indica lo scarto quadratico medio calcolato su tutta la popolazione in esame
- % : indica il divario medio percentuale tra la mia soluzione e l'ottimo
- $\%_{min}$: indica il divario percentuale tra la mia migliore soluzione e l'ottimo
- $\%_{max}$: indica il divario percentuale tra la mia peggiore soluzione e l'ottimo
- T : indica il tempo d'esecuzione in secondi; un * dopo il numero significa che il time-out è scaduto

6.2 Risultati attesi

Un algoritmo tabu search con intorno generato tramite 3–scambi o con intorno variabile non era mai stato utilizzato prima per la risoluzione del TSPDC.

Sono conosciute, invece, le potenzialità della tecnica 3-scambi nell'ambito della ricerca locale classica, nella quale l'algoritmo si arresta nel primo minimo locale individuato. In quest'ambito è noto che il 3-scambi produca risultati, normalmente, migliori o uguali a quelli del 2-scambi con rare eccezioni.

Eliminando dai miei algoritmi il meccanismo tabu search ho potuto effettuare alcuni test per verificare l'effettivo comportamento del 2-scambi e del 3-scambi in una ricerca locale semplice.

I risultati sono raccolti nella Tabella 6.1

Tabella 6.1 – Ricerca locale classica

	2 - scambi	**3 - scambi**	**Intorno variabile**
Mappa	Z_{locale}	Z_{locale}	Z_{locale}
V50d	460	442	442
V75d	576	572	572
V100d	521	520	515
E50e2	679	677	676
E75e2	758	758	758
E100e2	877	877	877
S50e2	143	140	140
S75e2	161	158	158
S100e2	205	183	183

Si evince facilmente dalla Tabella 6.1 che il 3-scambi fornisce risultati sempre migliori o uguali al 2 scambi. È dunque lecito aspettarsi che, inserendo il meccanismo tabù search, le soluzioni migliorino in modo simile per i due algoritmi. In particolare, detto *improve%* il miglioramento percentuale tra Z_{locale} e Z, in una particolare mappa, è logico attendere che esso sia simile per entrambi gli algoritmi o che, seppur migliore nel 2-scambi - il quale gode di un maggiore margine di miglioramento - Z sia generalmente migliore nel 3-scambi.

Nella tabella 6.1 è riportato inoltre il valore ottenuto dall'algoritmo con intorno variabile questo valore è uguale a quello ottenuto dal 3-scambi tranne in due casi nei quali è inferiore. L'algoritmo con intorno variabile, in un contesto di ricerca locale classica, migliora fino ad ottenere il risultato del 2-scambi a questo punto passa all'analisi degli intorni ottenuti con il 3-scambi e si interrompe quando ottiene il primo risultato non migliorante. Da questo algoritmo, implementando il meccanismo tabu search, ci si attende di ottenere dei risultati sicuramente migliori a quelli del 2-scambi e che, mediamente, apportino un miglioramento anche alle soluzioni del 3-scambi.

Per quanto riguarda i tempi, fare una previsione è più semplice. È noto, infatti, che il costo per l'esplorazione completa dell'intorno generato tramite 2-scambi è $O(N^2)$, mentre per il 3-scambi il costo è pari a $O(N^3)$. È, dunque, lecito attendersi che $\frac{T_{2-scambi}}{N(N-1)} \cong \frac{T_{3-scambi}}{N(N-1)(N-2)}$.

Una volta ottenuto il tempo del 2-scambi su una particolare mappa

$$T_{3\_scambi} \cong N(N-1)(N-2)\frac{T_{2-scambi}}{N(N-1)} \qquad \text{(Formula6.2)}$$

Per maggiore chiarezza riporto un esempio numerico.
Dato $T_{2\text{-}scambi} = 1.50$ s su una mappa con 51 nodi

$$T_{3\text{-}scambi} \cong (51 \times 50 \times 49)\frac{1.50}{(51 \times 50)} = 73.5 \text{ s}$$

Questa semplice formula ci permette di fornire solo una stima approssimativa del tempo impiegato dall'algoritmo 3-scambi. Il tempo reale d'esecuzione dipende, infatti, dal numero effettivo di soluzioni che l'algoritmo deve analizzare; questo numero non è costante, ma varia a seconda del numero di soluzioni etichettate come tabu.

Più complessa è la previsione riguardante i tempi dell'algoritmo con intorno variabile. È, comunque, possibile affermare che i tempi di questo algoritmo saranno più vicini a quelli del 3-scambi che a quelli del 2-scambi. Per supportare questa tesi è sufficiente pensare alla struttura dell'algoritmo. Esso, infatti, termina dopo aver eseguito prima 200 e successivamente 500 iterazioni non miglioranti ognuna delle quali prevede l'esplorazione completa di entrambi gli intorni generati dal 2-scambi e dal 3-scambi, con costo per ogni iterazione pari a $O(N^3)+ O(N^2)$. La velocità dell'algoritmo è bilanciata dal tempo richiesto per eseguire iterazioni non miglioranti e quindi è prevedibile che i tempi dell'algoritmo con intorno variabile siano inferiori a quelli dell'algoritmo con 3-scambi, ma non paragonabili, se non per rare eccezioni, a quelli dell'algoritmo con 2-scambi.

6.3 Risultati sperimentali

In questa sezione sono raccolti i risultati sperimentali riassunti in tre tabelle, una per ogni classe di problema.

Tabella 6.2 – Classe A

			2 - scambi						3 - scambi						Intorno variabile					
Mappa	β	Z^*	Z	σ	$\%$	$\%_{min}$	$\%_{max}$	T	Z	σ	$\%$	$\%_{min}$	$\%_{max}$	T	Z	σ	$\%$	$\%_{min}$	$\%_{max}$	T
V25b	0.05	306	307.25	6.18	4.22	0	1.66	0.26	315.8	3.47	2.51	0.98	4.57	3.88	306.1	0.31	0.03	0	0.32	3.1
V25c	0.10	308	315.4	4.14	2.40	0.32	3.90	0.26	324.8	0.97	5.45	5.19	5.44	3.65	311.9	3.61	1.26	0	2.92	2.97
V25d	0.20	313	325.5	11.1	3.99	0.31	11.1	0.25	322.15	6.10	2.92	0	4.79	3.57	314.6	0.8	0.51	0	0.63	2.8
V50b	0.05	426	435.55	5.38	2.24	0.23	4.69	2.67	446	6.19	4.69	1.87	7.27	49.22	434.85	5.22	2.77	0.23	4.92	39.13
V50c	0.10	426	436.3	4.68	2.4	0.70	4.46	1.81	444.95	6.68	4.44	0.93	7.27	49.85	432.65	4.71	1.56	0	4.22	37.83
V50d	0.20	426	435.6	4.80	2.23	0.46	4.22	1.74	438.15	5.49	2.82	0.46	4.22	47.12	433.8	4.16	1.83	0	3.28	37.93
V75b	0.05	538	554.55	3.38	3.07	1.67	4.64	7.6	567.15	6.78	5.41	4.27	7.8	585.16	553.9	4.76	2.95	1.67	4.08	491.61
V75c	0.10	539	557.3	4.32	3.95	2.04	4.82	6.11	562.35	4.97	4.33	2.98	5.38	565.63	554.65	4.36	2.90	1.66	4.45	438.57
V75d	0.20	539	557.8	3.24	3.48	2.04	4.45	6.19	563.25	2.87	4.49	3.52	5.19	559.86	551.35	3.88	2.91	1.29	3.33	451.91
V100b	0.05	501	513.1	8.92	2.41	0.59	5.78	19.98	513.3	6.74	2.45	1.19	4.79	1518.92	509.3	5.97	1.65	0.19	4.79	1057.74
V100c	0.10	501	515.85	12.1	2.96	0.39	8.38	20.12	514	6.77	2.59	1.19	4.79	1542.81	509.9	5.95	1.77	0.19	3.79	1133.38
V100d	0.20	502	511.35	7.59	1.86	0.39	4.78	19.98	512.1	5.68	1.99	0.7	4.78	1588.41	509.95	5.07	1.58	0.39	3.38	1089.98
V150b	0.05	698	743.5	12.6	6.51	3.01	9.88	64.86	743.95	11.72	6.95	4.58	9.31	3600*	737.05	10.66	5.59	2.72	7.59	3600*
V150c	0.10	699	742.05	7.70	6.15	4.43	8.15	63.61	742.05	6.43	6.15	4.57	7.15	3600*	739.65	9.26	5.81	3.01	7.29	3600*
V150d	0.20	702	740.05	11.5	5.42	3.56	7.11	64.99	740.8	6.37	5.52	4.41	7.12	3600*	735.5	8.12	4.77	2.42	6.12	3600*
V199b	0.05	761	812.35	9.89	6.74	4.33	8.93	125.5	809.05	7.39	6.31	4.33	7.75	3600*	809.65	8.20	6.39	3.94	7.22	3600*
V199c	0.10	761	808.1	12.6	6.18	3.28	8.80	121.2	808.35	7.80	6.22	3.54	7.22	3600*	806.76	8.04	6.01	2.89	7.62	3600*
V199d	0.20	762	808.35	13.1	6.08	3.14	8.92	130.1	808.4	8.97	6.08	3.41	7.48	3600*	807.07	9.75	5.91	2.75	7.48	3600*
Media				7.95	4.01	1.71	6.37	36.51		6.18	4.51	2.67	6.24	1562.11		5.71	3.12	1.29	4.63	1465.94

Tabella 6.3 – Classe B

			2 - scambi						3 - scambi						Intorno variabile					
Mappa	β	Z^*	Z	σ	%	$\%_{min}$	$\%_{max}$	T	Z	σ	%	$\%_{min}$	$\%_{max}$	T	Z	σ	%	$\%_{min}$	$\%_{max}$	T
E25b2	0.05	462	464.6	3.47	0.56	0	1.73	0.31	481.65	12.95	4.25	0.64	8.07	4.55	464.5	2.49	0.54	0	1.08	3.31
E25c2	0.10	462	470.75	7.44	1.89	0	3.89	0.29	469	0	1.51	1.51	1.51	3.95	463	2.03	0.2	0	1.08	3.45
E25d2	0.20	462	470.45	5.39	1.82	0	3.24	0.29	495.9	8.24	7.33	5.62	9.30	4.21	462.3	1.33	0.06	0	1.29	3.95
E25e2	∞	460	464	3.83	0.86	0	2.39	0.29	645.75	2.20	1.25	0.43	1.52	4.12	462	0	0.43	0.43	0.43	3.28
E50b2	0.05	599	612.85	14.54	2.31	0	9.34	1.41	637.9	21.8	6.4	0.66	10.51	67.2	605.25	7.93	1.04	0	3.67	46.11
E50c2	0.10	599	610.65	10.19	1.94	0.16	5.17	1.35	622.1	13.17	3.85	1.50	8.18	60.9	610.45	10.45	1.91	0	4.84	44.99
E50d2	0.20	599	623.7	17.40	4.12	0	7.67	1.39	633.05	7.84	5.68	4	7.51	61.8	609.6	12.32	1.76	0	5.84	45.74
E50e2	∞	612	646.3	18.75	5.60	0	9.31	1.31	658.1	11.79	7.53	4.41	11.1	59.20	628.75	11.80	2.73	0	6.69	44.52
E75b2	0.05	658	706.65	15.26	7.39	2.88	10.79	7.8	698.3	14.76	6.12	0.15	10.33	579.8	692.7	17.22	5.27	0.30	9.87	401.
E75c2	0.10	659	702.25	10.2	6.56	1.82	9.55	8.05	706.3	5.46	7.17	5.61	8.04	564.45	700.45	3.04	6.28	5.61	7.28	382.32
E75d2	0.20	663	711.8	14.92	7.36	1.80	10.40	7.1	705.95	5.43	6.47	5.58	7.99	504.85	690.95	8.50	4.24	1.80	4.42	395.3
E75e2	∞	707	749.65	18.95	6.03	2.82	12.16	6.84	740	15.82	4.66	1.83	8.76	490.21	727.7	10.39	2.92	0.56	5.09	369.83
E100b2	0.05	797	837.8	18.92	5.11	1.12	8.65	15.46	840.2	15.30	5.42	2.38	7.52	1347.76	816.71	7.30	2.47	1	3.51	850.69
E100c2	0.10	807	849.75	12.09	5.29	2.35	8.92	13.53	864	13.13	7.06	4.58	10.4	1318.41	837.7	12.35	3.80	0	6.31	839.98
E100d2	0.20	811	863.55	14.59	6.47	4.19	10.23	13.6	882	12.01	8.75	5.91	10.35	1312.85	848.95	12.6	4.67	2.21	6.78	858.37
E100e2	∞	805	846.68	10.6	5.19	2.85	7.08	12.34	856.15	10.40	6.35	4.0	7.95	1383.92	833	10	3.47	0.99	5.59	907.74
E150b2	0.05	902	973.7	19.25	7.9	4.32	12.19	65.27	961.88	18.69	6.63	2.99	9.75	3600*	955.8	9.89	5.96	4.1	7.31	3568
E150c2	0.10	906	954.7	11.93	5.37	2.2	8.38	63.13	956.55	19.64	5.58	2.2	8.16	3600*	944.2	14.57	4.21	1.65	6.29	3600*
E150d2	0.20	919	968.45	17.76	5.38	2.39	10.11	64.07	961.75	16.85	4.65	1.08	6.42	3600*	948.87	15.29	3.25	1.08	6.42	3600*
E150e2	∞	901	962.95	18.85	6.87	3.10	9.87	68.81	960.35	18.35	6.58	3.1	8.76	3600*	945.7	17.32	4.96	1.77	8.54	3600*
E200b2	0.05	1105	1173.66	19.01	6.21	4.25	11.31	122.7	1159.6	8.71	4.94	3.16	6.15	3600*	1159.8	11.61	4.95	2.35	6.06	3600*
E200c2	0.10	1110	1173.05	15.82	5.68	3.78	10.81	128.9	1158.85	8.55	4.40	2.70	5.67	3600*	1157.51	10.92	4.27	2.52	5.76	3600*
E200d2	0.20	1109	1173.15	15.83	5.78	3.87	10.91	128.4	1159.6	8.73	4.56	2.79	5.77	3600*	1156.76	11.67	4.30	2.34	5.95	3600*
E200e2	∞	1106	1165.05	12.85	5.33	2.89	6.96	130.1	1157.74	9.40	4.67	3.07	5.96	3600*	1155.25	12.17	4.45	2.26	6.23	3600*
Media				13.66	4.87	1.94	8.37	35.94		11.63	5.49	2.91	7.73	1523.6		9.71	3.25	1.29	5.25	1415.3

Tabella 6.4 – Classe C

			2 - scambi						3 - scambi						Intorno variabile					
Mappa	β	Z^*	Z	σ	%	$\%_{min}$	$\%_{max}$	T	Z	σ	%	$\%_{min}$	$\%_{max}$	T	Z	σ	%	$\%_{min}$	$\%_{max}$	T
S25b2	0.05	233	238.15	2.59	2.21	0	3	0.19	238.7	2.1	2.44	0	3	4.41	236.85	3.56	1.6	0	3.43	3.51
S25c2	0.10	233	240.2	4.27	3.09	0.42	5.57	0.19	239.25	1.78	2.68	0.85	3	4.74	237.8	3.86	2.06	0	5.57	3.21
S25d2	0.20	240	240.2	0.4	0.08	0	0.41	0.2	242.05	1.68	0.85	0	1.66	4.32	240.6	0.91	0.25	0	0.83	3.18
S25e2	∞	233	242.85	6.18	4.22	1.71	8.15	0.17	246.45	1.62	5.77	5.15	6.86	4.24	237.75	4.33	2.03	0	5.15	3.34
S50b2	0.05	135	138.6	2.15	2.66	0	5.18	1.51	137.15	1.10	1.59	0	3.73	87.2	137.85	1.31	2.11	0.74	3.70	56.85
S50c2	0.10	135	139.25	1.21	3.15	2.22	5.18	1.49	138.05	1.43	2.25	0	3.70	70.31	138.3	1.31	2.44	0.74	3.70	53.5
S50d2	0.20	135	139	2.77	2.96	0	6.66	1.49	139.35	1.65	3.22	1.48	4.44	70.01	139.15	1.58	3.07	0.74	4.44	51.51
S50e2	∞	135	139.2	1.43	3.11	1.48	4.44	1.46	137.65	1.38	1.96	0.74	3.70	68.22	137.55	1.39	1.88	0	4.44	51.7
S75b2	0.05	150	156.5	1.87	4.33	1.33	6.66	7.81	154.3	1.01	2.86	2	4	655.66	153.9	1.98	2.6	0.66	4.66	359.51
S75c2	0.10	150	157.75	2.78	5.16	2	8	7.85	156.3	1.84	4.2	2.66	6	630.93	153.71	1.82	2.47	0.66	4	361.3
S75d2	0.20	150	158.6	2.14	5.73	2.66	7.33	7.2	156	0.89	4	3.33	4.66	631.41	156.55	1.80	4.36	2.66	6	358.74
S75e2	∞	150	157.05	2.57	4.27	2.66	8	7.34	153.7	0.9	2.46	1.33	3.33	602.11	154.	1.53	3	1.33	5.33	342.18
S100b2	0.05	173	181.85	2.07	5.11	2.89	7.51	18.7	183.55	2.15	6.09	4.04	8.09	1355.44	180.7	2.10	4.45	1.73	5.78	881.84
S100c2	0.10	173	182.85	2.83	5.69	1.73	10.40	17.34	182.35	2.32	5.40	4.04	7.51	1381.83	181.5	3.12	4.9	1.73	7.5	887.19
S100d2	0.20	173	182.65	2.85	5.57	2.89	9.24	17.2	180.1	1.86	4.10	2.31	5.78	1399.76	179.98	2.21	4.01	2.31	6.35	916.13
S100e2	∞	173	184.6	3.16	6.70	3.46	11.56	16.9	180.55	2.63	4.36	1.15	6.93	1439.44	181.25	2.34	4.76	2.89	8.09	871.82
S150b2	0.05	82	90.5	2.39	10.3	6.09	19.51	29.86	90.95	2.43	10.91	7.31	15.85	3600*	89.53	2.09	9.18	4.87	13.41	3600*
S150c2	0.10	82	91.75	1.94	11.89	8.53	15.85	30.01	91.21	1.81	11.23	8.53	14.63	3600*	90.57	1.84	10.45	7.31	13.41	3600*
S150d2	0.20	82	94.1	3.42	14.75	4.87	24.39	30.38	92	0	12.19	12.19	12.19	3600*	90.5	2.5	10.36	4.87	14.63	3600*
S150e2	∞	82	91.35	2.19	11.40	7.31	18.29	30.59	89.85	2.29	9.58	3.65	9.58	3600*	88.62	1.72	8.07	4.87	9.75	3535.22
S200b2	0.05	61	61.75	1.23	1.22	0	8.19	107.44	61.85	0.85	1.39	0	4.91	3600*	61.85	1.16	1.39	0	6.55	3600*
S200c2	0.10	61	61.8	0.4	1.31	0	1.63	105.81	61.66	0.81	1.09	0	3.27	3600*	61.7	0.45	1.15	0	1.63	3600*
S200d2	0.20	61	61.65	0.91	1.06	0	4.91	105.09	61.9	1.39	1.47	0	3.27	3600*	61.65	0.66	1.06	0	3.27	3600*
S200e2	∞	61	61.65	0.56	1.06	0	3.27	112.21	62.33	0.47	2.18	1.63	3.27	3600*	61.33	0.47	0.54	0	1.63	3600*
Media				2.26	4.87	2.17	8.47	27.43		1.51	4.34	2.59	5.97	1550.41		1.91	3.67	1.58	5.96	1414.19

6.4 Analisi dei risultati ottenuti

Considerati i risultati attesi e le premesse discusse nella sezione 6.2, ottenuti i risultati sperimentali, è possibile effettuarne un'analisi ed un confronto sia per quanto concerne la qualità delle soluzioni prodotte sia per i tempi impiegati dagli algoritmi.

Nella Tabella 6.5 è riportato il confronto tra i dati della Tabella 6.1 e i risultati sperimentali ottenuti sulle medesime mappe utilizzate per la ricerca locale. L'attenzione della tabella si è focalizzata sul miglioramento percentuale (*improve%*) tra ricerca locale classica e tabu search.

Tabella 6.5 – Confronto ricerca locale vs tabu search

		2 - scambi			**3 - scambi**			**Intorno variabile**		
Mappa	*Z**	Z_{locale}	*Z*	*improve%*	Z_{locale}	*Z*	*improve%*	Z_{locale}	*Z*	*improve%*
V50d	426	460	435.6	5.30	442	438.15	0.87	442	433.8	1.85
V75d	539	576	557.8	3.15	572	563.25	1.52	572	551.35	3.61
V100d	502	521	511.35	1.85	520	512.1	1.51	515	509.95	0.98
E50e2	612	679	646.3	4.81	677	658.1	2.79	676	628.75	6.98
E75e2	707	758	749.65	1.10	758	740	2.37	758	727.7	3.99
E100e2	803	877	846.68	3.45	877	856.15	2.37	877	833	5.01
S50e2	135	143	139.2	2.65	140	137.65	1.67	140	137.55	1.75
S75e2	150	161	157.5	2.17	158	153.7	2.72	158	154	2.53
S100e2	173	205	184.6	9.95	183	180.55	1.33	183	181.25	0.95
Media				3.83			1.91			3.07

Come è facilmente deducibile dalla Tabella 6.5 le aspettative riguardo all'algoritmo con 3-scambi sono state disattese. Infatti, non solo *improve%* del 3-scambi è minore di *improve%* del 2-scambi ma, $Z_{3\text{-}scambi}$ è in più del 50% dei casi peggiore rispetto a $Z_{2\text{-}scambi}$. L'algoritmo con intorno variabile ha *improve%* di poco inferiore a quella del 2-scambi ma, offre soluzioni medie sempre migliori. È da sottolineare come rare volte capiti che l'algoritmo 3-scambi produca soluzioni migliori di quello con intorno variabile e in questi casi entrambi questi algoritmi superino il 2-scambi. Questi risultati sono ulteriormente confermati da quelli riportati delle Tabelle 6.2, 6.3 e 6.4.

Un quadro generale della qualità delle soluzioni prodotte dai tre diversi algoritmi è riportato nella tabella 6.6.

Tabella 6.6

	2 - scambi			**3 - scambi**			**Intorno variabile**		
Classe del problema	%	$\%_{min}$	$\%_{max}$	%	$\%_{min}$	$\%_{max}$	%	$\%_{min}$	$\%_{max}$
Classe A	4.01	1.71	6.37	4.51	2.67	6.24	3.12	1.29	4.63
Classe B	4.87	1.94	8.37	5.49	2.91	7.73	3.25	1.29	5.25
Classe C	4.87	2.17	8.47	4.34	2.59	5.97	3.67	1.58	5.96
Media	4.58	1.94	7.73	4.78	2.72	6.64	3.34	1.38	5.28

Nella Tabella 6.6 sono riportate le medie dei gap percentuali medi, minimi e massimi rispetto alle soluzioni ottime trovate da Baldacci in[9]. Si nota che per la Classe A e B di problemi il 2-scambi produce soluzioni medie e minime migliori del 3-scambi il quale però ha il pregio di offrire soluzioni massime migliori rispetto. Al contrario nella Classe C la situazione è a favore del 3-scambi tranne che per le soluzioni minime. Nonostante ciò, facendo la media dei risultati ottenuti, si evince che il 2-scambi produce soluzioni medie e minime migliori mostrando però una maggiore variabilità delle soluzioni prodotte. L'algoritmo con intorno variabile si dimostra il migliore in tutte le classi di problemi fornendo soluzioni medie migliori di più dell'1% rispetto al 2-scambi ed evidenziando una minore variabilità delle soluzioni.

Analizzando lo scarto quadratico medio dei tre algoritmi è possibile trarre delle considerazioni sulla robustezza degli stessi.

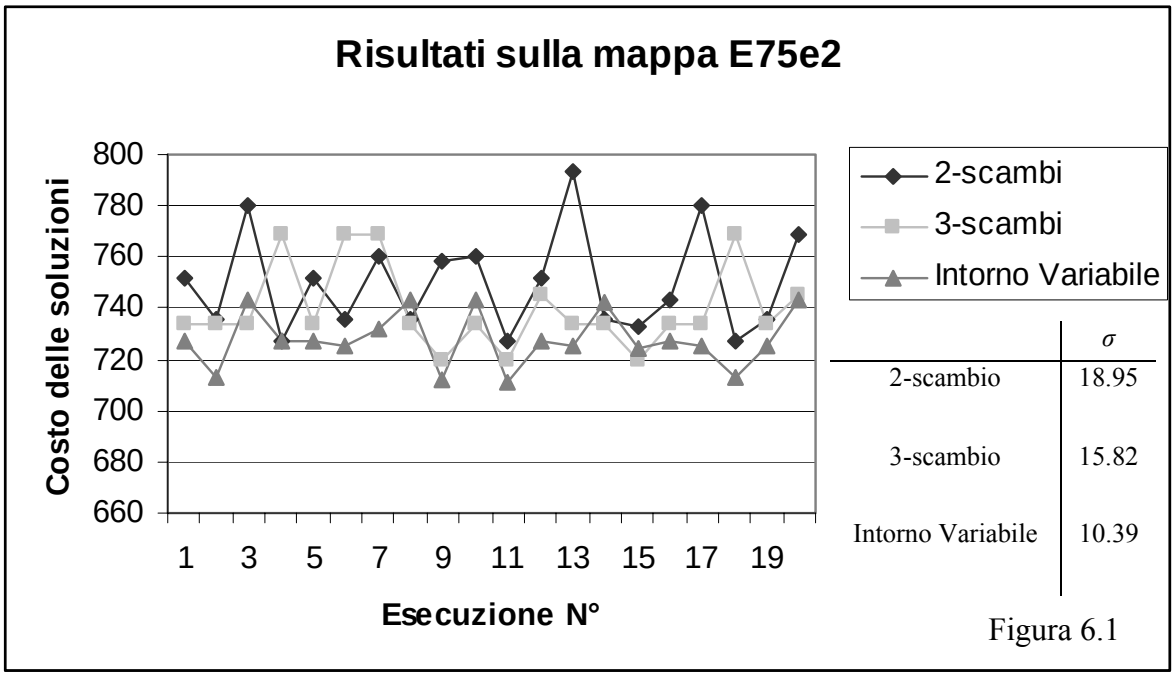
La media del σ e del $\sigma\%$ (scarto quadratico medio percentuale) divisa per classi la è riportata nella Tabella 6.7.

Tabella 6.7

	2 - scambi		**3 - scambi**		**Intorno variabile**	
Classe del problema	σ	$\sigma\%$	σ	$\sigma\%$	σ	$\sigma\%$
Classe A	7.95	1.41	6.18	1.09	5.71	1.02
Classe B	13.66	1.71	11.63	1.43	9.71	1.23
Classe C	2.26	1.55	1.51	1.04	1.91	1.31
Media		1.56		1.18		1.18

L'algoritmo 2-scambi evidenzia lo scarto quadratico medio maggiore in ogni classe di problema.L'algoritmo con intorno variabile ha σ minore per le Classi A e B mentre nella Classe C è l'algoritmo 3-scambi ad avere prestazioni migliori.

L'elevata σ è sintomo di scarsa robustezza dell'algoritmo il quale evidenzia, se si dispongono le soluzioni in uno spazio cartesiano, dei picchi che identificano soluzioni o molto buone o pessime. In Figura 6.1 è presente il grafico delle soluzioni ottenute sulla mappa E75e2 ed una tabella che evidenzia il σ dei tre algoritmi in quella mappa.



	σ
2-scambio	18.95
3-scambio	15.82
Intorno Variabile	10.39

Figura 6.1

L'algoritmo 2-scambi, com'evidenzia la Figura 6.1, è quello che produce le soluzioni più distanti tra loro. Esso, infatti, genera numerosi picchi che delineano soluzioni completamente fuori dalla media. Nel caso riportato in Figura 6.1 è facile notare la maggiore robustezza degli algoritmo 3-scambi e con intorno variabile; quest'ultimo, in particolare, evidenzia una scarsa dispersione delle soluzioni accompagnata ad un'ottima qualità dei risultati ottenuti.

Con gli elementi in nostro possesso possiamo quindi trarre un quadro complessivo della qualità delle soluzioni prodotte. Ne risulta che il tabu search con intorno variabile offre, tranne che in pochi casi, non solo soluzioni medie migliori, ma anche soluzioni minime e massime migliori rispetto agli altri algoritmi. Il miglioramento medio rispetto al 2-scambi è quantificabile in circa 1-2%. Il tabu search con intorno variabile si dimostra inoltre più solido ed affidabile grazie ad una minore dispersione delle soluzioni e dunque ad uno scarto quadratico medio inferiore del 24% rispetto al tabu search con 2-scambi.

Seppur dimostri maggiore solidità rispetto al 2-scambi, il tabu search con 3-scambi offre una qualità delle soluzioni altalenante, che risulta essere inferiore a quella del 2-scambi nella maggioranza dei casi.

Prima di considerare i tempi d'esecuzione è opportuno analizzare le cause che hanno portato il tabu search con 3-scambi ad offrire soluzioni che non corrispondono alle aspettative. Io ritengo che il meccanismo tabu search, così come è stato implementato in questi algoritmi, sia troppo limitante per il 3-scambi. In particolare, analizzando lo stato della tabu list durante l'esecuzione dell'algoritmo, è possibile notare come questa sia riempita e svuotata molto più velocemente rispetto agli altri due algoritmi. Questo implica che l'esplorazione dell'intorno prodotto dal 3-scambi sia inefficiente. Si deve infatti considerare che la "fame di archi" del 3-scambi è il 50% maggiore di quella del 2-scambi e che ad ogni iterazione il 3-scambi introduce nella tabu list 3 archi invece dei 2 del 2-scambi. L'algoritmo si ritrova -in breve tempo- condotto dalle restrizioni imposte dalla tabu search, ad ottenere un minimo locale ed a questo punto il rapido svuotamento della tabu list impedisce all'algoritmo di allontanarsi, in modo sufficiente, dalla soluzione alla quale vuole sfuggire. Analizzando le tabelle 6.2, 6.3, 6.4, è possibile notare che le soluzioni prodotte dal 3-scambi tendono a migliorare, sia come qualità sia come scarto quadratico medio, all'aumentare del numero di nodi. Questo comportamento risulta logico,alla luce di quanto detto poco sopra ed infatti, aumentando il numero dei nodi, aumentano le possibilità di scelta dell'algoritmo e si mitigano gli effetti dei due comportamenti indesiderati descritti.

Analizzando i tempi di esecuzione degli algoritmi riportati nelle tabelle 6.2, 6.3, 6.4 è possibile, utilizzando la Formula 6.2, realizzare la Tabella 6.8, la quale riporta un confronto tra i tempi stimati con la Formula 6.2 e i tempi di esecuzione reali sulle mappe della Tabella 6.1.

La colonna Gap % riporta il divario percentuale tra tempo stimato e tempo reale.

Tabella 6.8

Mappa	$T_{2\text{-}scambi}$	$T_{3\text{-}scambi}$ *Reale*	$T_{3\text{-}scambi}$ *Stimato*	*Gap %*
E50e2	1.31	62.7	64.19	-8.42
E75e2	6.84	490.21	506.16	-3.25
E100e2	14.34	1383.92	1419.66	11.72
V50d	1.74	47.12	85.26	-80.94
V75d	6.19	559.86	458.06	18.18
V100d	19.98	1588.41	1978.02	-24.52
S50e2	1.46	68.22	71.54	-4.86
S75e2	7.34	602.11	543.16	9.79
S100e2	16.9	1439.44	1673.1	-16.23
Media	8.45	693.55	755.46	-10.95

La Formula 6.2 produce stime che sono mediamente pari al 110.95% del tempo effettivo impiegato dall'algoritmo. Inoltre se si considerano i valori assoluti della colonna *Gap %* si ottiene una media pari al 19.77%. In definitiva è possibile dire che questa formula può essere usata per effettuare delle stime sommarie, ma certamente non per delle previsioni precise.

Nella Tabella 6.9 sono riportate le medie divise per Classe di problema dei tempi d'esecuzione dei tre algoritmi escludendo le mappe sulle quali l'esecuzione è stata sospesa per scadenza del time-out.

Tabella 6.9

Classe del problema	$T_{2\text{-}scambi}$	$T_{3\text{-}scambi}$	$T_{IntornoVariabile}$
Classe A	7,24	543,17	398,91
Classe B	5,71	485,51	325,03
Classe C	6,69	525,62	325,34
Media	6,54	518,1	349,76

Confrontando questi tempi si nota immediatamente che il $T_{3\text{-}scambi}$ è circa 80 volte il $T_{2\text{-}scambi}$ ed è pari a 3/2 del $T_{IntornoVariabile}$. L'algoritmo con intorno variabile risulta, come da previsione, avere tempi inferiori, circa i 2/3, di quelli della tabu search con 3-scambi, ma non paragonabili a quelli del 2-scambi. Infatti il $T_{IntornoVariabile}$ risulta pari a circa 55 $T_{2\text{-}scambi}$.

L'analisi dei tempi ha confermato praticamente tutte le aspettative senza produrre sorprese. I tempi degli algoritmi sono strettamente legati al metodo di esplorazione dell'intorno e dunque potrebbero essere abbassati utilizzando politiche d'esplorazione differenti dalla best improve, la quale prevede un'analisi completa dell'intorno generato. È impossibile, vista la differente grandezza dell'intorno che un algoritmo 3-scambi ottenga tempi simili ad un 2-scambi che utilizza la medesima politica d'esplorazione dell'intorno.

7. Conclusione

Dai test effettuati risulta che il tabu search con intorno variabile offre le soluzioni qualitativamente migliori. Esso si dimostra essere l'algoritmo più robusto, fornisce risultati migliori di entrambi gli altri algoritmi e lo fa in tempi più brevi rispetto al 3-scambi.

Il tabu search con 2-scambi è, tra gli algoritmi analizzati, quello più indicato se si vuole una discreta soluzione in tempi molto brevi. Bisogna, tuttavia, considerare che esso offre una scarsa robustezza e quindi può richiedere diverse esecuzioni prima di produrre una buona soluzione.

Il tabu search con 3-scambi invece ha dimostrato di non soddisfare le aspettative, ed oggi, senza una profonda ristrutturazione del meccanismo tabu, non ha motivo di essere adottato poiché non fornisce soluzioni migliori ed ha i peggiori tempi tra tutti gli algoritmi considerati.

Confrontando i risultati da me ottenuti con i risultati di Baldacci, Hadjicostantinou e Mingozzi in[9], si nota che l'approccio tabu search al TSPDC offre soluzioni che distano dall'ottimo pochi punti percentuali con tempi che sono spesso paragonabili.

La ricerca tabu è in grado di offrire dunque una alternativa all'utilizzo di complessi e costosi LP-solver commerciali, come ad esempio il CPLEX prodotto da ILOG, utilizzato da Baldacci. Essa rinunciando alla certezza dell'ottimalità riesce a produrre risultati convincenti in tempi accettabili; in particolare tramite tabu search si riesce a generare in modo semplice ed a basso costo soluzioni che distano mediamente il 3% dall'ottimo.

Restano aperte numerose strade per future ricerche. Infatti l'algoritmo con intorno varabile offre buone soluzioni, ma con tempi che, se paragonati a tabu search con 2-scambi, sono molto lunghi. Si potrebbero studiare nuovi metodi di esplorazione degli intorni generati che offrano soluzioni buone in tempi brevi. Infine, si potrebbe implementare un nuovo meccanismo tabu search per l'algoritmo con 3-scambi che sia meno restrittivo e che metta in luce le buone potenzialità dimostrate da questo algoritmo nella ricerca locale classica.

Bibliografia

[1] Gutin G., Punnen A. P. “Travelling salesman problem and its variations (Combinatorial Optimization V.12)”, Kluver 2002.

[2] Mosheiov G. “The travelling salesman problem with pick-up and delivery”. European Journal of Operational Research 1994; 79: 299-310.

[3] Anily S., Mosheiov G. “The travelling salesman problem with delivery and backhauls”. Operational Research Letters 1994; 16: 11-8.

[4] Gendreau M., Hertz A., Laporte G. “The travelling salesman problem with backhauls”. Computers & Operations Research 1996; 23: 501-8.

[5] G. Pesant, M. Gendreau, and J.-M. Rousseau. GENIUS-CP: “A generic single-vehicle routing algorithm”. In Proceedings of CP '97, pages 420--433. Springer-Verlag, 1997.

[6] Halse K. “Modelling and solving complex vehicle routing problems”. PhD Thesis, no. 60; IMSOR, The Technical University of Denmark, 1992.

[7] Tzoreff T.E., Granot D., Granot F., Sosic G., "The vehicle routing problem with pickups and deliveries on some special graphs", Discrete Applied Mathematics 2002; 116: 193-229.

[8] Gendreau M., Laporte G., Vigo D. “Heuristics for the travelling salesman problem with pickup end delivery”. Computers & Operations Research 1999; 26: 699-714.

[9] Baldacci R., Hadjicostantinou E., Mingozzi A. “An exact algorithm for the traveling salesman problem with deliveries and collections”. Networks 2003; 42: 26-41

[10] Aarts, Emile H. L. Lenstra, Jan Karel (eds.) "Local search in combinatorial optimization” Chichester [England] ; New York : Wiley, 1997.

[11] Glover F., Laguna M. Tabu Search. Norwell, MA: Kluwer, 1997.