Indice

Introduzione

Negli ultimi decenni si è verificato un crescente utilizzo di pacchetti software basati su tecniche di ricerca operativa e programmazione matematica per la gestione efficiente della fornitura di beni e servizi nei sistemi di distribuzione.

Il grande numero di applicazioni reali, sia nel Nord America sia in Europa, ha ampiamente dimostrato che l'utilizzo di software per la pianificazione dei processi di distribuzione produce un sostanziale risparmio (generalmente dal 5% al 20%) nei costi globali di trasporto. È facile osservare come l'impatto di questo risparmio sul sistema economico di un Paese sia significativo dal momento che i processi di trasporto riguardano tutte le fasi della produzione dei beni ed i costi relativi rappresentano una componente rilevante (generalmente dal 10% al 20%) del costo finale.

Il successo nell'utilizzo di tecniche di ricerca operativa è dovuto non solo allo sviluppo hardware e software nel campo dell'informatica e alla crescente integrazione dei sistemi informativi nel processo produttivo ed in quello commerciale ma soprattutto allo sviluppo di nuovi modelli che cercano di prendere in considerazione tutte le caratteristiche dei problemi reali ed alla concezione di nuovi algoritmi che permettono di trovare buone soluzioni in tempi di calcolo accettabili.

Recentemente le questioni ambientali stanno ricevendo grosse attenzioni: a livello mondiale si sta cercando di orientare la legislazione e le politiche dei paesi alla salvaguardia dell'ambiente e si stanno facendo considerevoli investimenti nelle attività

di recupero e riciclaggio di materiali di scarto. Questo porta a modellare problemi in cui si ha l'integrazione del flusso principale di beni che sono prodotti e distribuiti con un flusso inverso di materiali di scarto che devono essere raccolti, immagazzinati, disassemblati e riciclati; porta cioè a considerare modelli di problemi che si collocano nell'ambito della logistica inversa; problemi che devono essere risolti in maniera ottima ai fini di rendere il riciclo di materiali di scarto più economico della loro distruzione. In particolare l'obiettivo della logistica inversa della distribuzione è quello di ottimizzare la consegna di merce dai depositi ai clienti e la contemporanea raccolta di materiale di scarto che dai clienti deve essere trasportato ai depositi o a impianti di riciclaggio specializzati. In quest'ottica, problemi classici come il Traveling Salesman Problem (TSP) e il Capacitated Vehicle Routing Problem (CVRP) devono essere riformulati.

Il CVRP consiste nell'ottimizzare i percorsi di un insieme di veicoli di una data capacità che devono consegnare merce a un insieme di clienti viaggiando lungo la rete di trasporto esistente. L'operazione relativa alla raccolta di materiale di scarto può essere incorporata nel CVRP in differenti modi: in [15]e[16] viene preso in considerazione il VRP with Backhauls (VRPB) in cui ogni veicolo può sia consegnare merce sia raccogliere materiale di scarto ma deve effettuare tutte le consegne prima di poter caricare materiale di scarto; in [3] Mosheiov descrive algoritmi euristici per il VRP with Simultaneous Pick-up and Delivery (VRPSPD) in cui, rispetto al VRPSB, non esiste un vincolo di precedenza tra l'ultima consegna e la prima operazione di raccolta; in [17] si considera infine il VRP with Pick-up and Delivery (VRPPD) in cui, rispetto al VRPSPD, i punti di prelievo della merce e di consegna del materiale di scarto non coincidono necessariamente con il deposito.

In questa tesi è preso in esame il VRPSPD sia nel caso in cui la domanda dei

clienti è divisibile sia nel caso in cui non lo è: nel capitolo 1 sono descritte le due formulazioni del problema, nei capitoli successivi vengono presentati nuovi algoritmi di approssimazione che possono essere utilizzati per risolvere il problema. Nel capitolo 2 vengono definiti quattro algoritmi che appartengono alla classe delle euristiche a due fasi in cui il problema viene risolto in due fase distinte: il raggruppamento (*clustering*) dei clienti in insiemi e la costruzione di un percorso (*route*) che permetta di visitare i clienti che appartengono ad un insieme. Dei quattro algoritmi definiti viene riportata l'analisi delle prestazioni nel caso peggiore ed una valutazione sperimentale comparata che dimostra tra l'altro come essi garantiscano prestazioni migliori di quelle di simili algoritmi proposti da Mosheiov in [3]. Nel capitolo 3 si definiscono intorni per la costruzione di efficienti algoritmi di ricerca locale; il loro funzionamento si basa sull'idea di migliorare una soluzione ammissibile eseguendo una sequenza di scambi di archi o spigoli oppure di vertici o nodi all'interno del singolo percorso o tra percorsi diversi. I risultati dei test effettuati permettono di confrontare le prestazioni degli algoritmi e la loro robustezza rispetto all'inizializzazione. Nel capitolo 4 viene infine presentato un tabu search basato sugli intorni definiti nel capitolo precedente. Il tabu search riceve in ingresso una soluzione ammissibile x_1, passa iterativamente dalla soluzione x_t alla soluzione x_{t+1} che appartiene all'intorno $N(x_t)$ e termina al verificarsi di determinate condizioni; se $f(x)$ rappresenta il costo di x, allora $f(x_{t+1})$ non necessariamente è minore di $f(x_t)$, di conseguenza occorre prestare particolare attenzione alla possibilità che l'algoritmo cicli su un insieme di soluzioni. I risultati dei test svolti mostrano come il tabu search sia l'algoritmo che garantisca il calcolo delle migliori soluzioni a fronte di tempi di calcolo accettabili; per avere una misura significativa dell'efficienza di questo algoritmo, ulteriori test sono stati poi eseguiti confrontando le sue prestazioni con quelle di un MIP (Mixed Integer Programming)

solver.

Gli algoritmi sono stati scritti in C standard, tutti i test sono stati eseguiti su un PC con processore Intel-Pentium 4 1.60 GHz, dotato di 640 MB di memoria RAM, sistema operativo Microsoft Windows 2000 Professional. Il software commerciale utilizzato per effettuare parte dei test del capitolo 4 è ILOG CPLEX 6.5.3.

Capitolo 1

VRPSPD: formulazione e metodi risolutivi

1.1 Vehicle Routing Problems

I problemi inerenti la distribuzione di beni materiali tra un deposito o un insieme di depositi e i clienti sono generalmente noti con il nome di Vehicle Routing Problems (VRPs) o Vehicle Scheduling Problems.

Distribuire beni significa servire un insieme di clienti grazie a veicoli che sono localizzati in uno o più depositi e che effettuano i loro spostamenti utilizzando la rete stradale esistente. La soluzione di un VRP è quindi rappresentata da un insieme di percorsi che iniziano e terminano nei depositi, ognuno dei quali deve essere effettuato da un unico veicolo. Tali percorsi devono essere determinati in maniera tale che tutte le richieste dei clienti siano esaudite, tutti i vincoli operazionali siano soddisfatti ed il costo globale del trasporto sia minimizzato.

Capacitated Vehicle Routing Problem

Tra i vehicle routing problems (VRPs) quello più semplice e il maggiormente studiato è il Capacitated VRP (CVRP).

Nella versione base del CVRP tutti i clienti sono caratterizzati dal medesimo tipo

di domanda, di consegna (*delivery demand*) o di raccolta (*pick-up demand*), il cui valore è deterministico e conosciuto a priori, gli h veicoli utilizzati per visitare i clienti sono identici (in particolare hanno tutti la stessa capacità Q) e partono da un unico deposito.
Risolvere il CVRP significa trovare un insieme di h percorsi di costo minimo (dove il costo di un percorso rappresenta la sua lunghezza o il tempo necessario ad effettuarlo) in modo che le seguenti condizioni siano verificate:

- ogni veicolo durante il suo percorso deve visitare il deposito;
- ogni cliente deve essere servito da un unico veicolo (non è possibile soddisfare parte della domanda del cliente quando lo si visita);
- la somma delle domande dei clienti visitati all'interno di un percorso non deve superare la capacità Q del veicolo.

Il Traveling Salesman Problem (TSP) è un caso particolare del CVRP e lo si ottiene nel caso in cui h sia uguale a 1 e Q sia maggiore o uguale all'ammontare complessivo delle domande dei clienti; essendo il TSP un problema di ottimizzazione NP-hard in senso forte ne consegue che anche il CVRP è un problema di ottimizzazione NP-hard in senso forte.

VRP with Pick-up and Delivery

Il VRP with Pick-up and Delivery (VRPPD) è una generalizzazione del CVRP.

Nella versione base, per ogni cliente i si ha che d_i e p_i ne rappresentano rispettivamente la *delivery demand* e la *pick-up demand*, O_i rappresenta l'origine (il deposito o un altro cliente) da cui prelevare i beni per soddisfare la *delivery demand* e D_i rappresenta la destinazione (il deposito o un altro cliente) dei beni che sono stati prelevati

soddisfando la *pick-up demand*. Nel VRPPD si assume che durante la visita a un cliente venga soddisfatta per prima la *delivery demand* e successivamente la *pick-up demand*; il carico di un veicolo prima di raggiungere un certo cliente è quindi pari al valore del carico iniziale meno l'ammontare delle domande di *delivery* soddisfatte più l'ammontare delle domande di *pick-up* soddisfatte.
Risolvere il VRPPD significa trovare un insieme di h percorsi di costo minimo in modo che le seguenti condizioni siano verificate:

- ogni veicolo durante il suo percorso deve visitare il deposito;
- ogni cliente deve essere visitato da un unico veicolo;
- il carico corrente del veicolo lungo il percorso non può eccedere la capacità Q del veicolo;
- per ogni cliente i, se O_i identifica un altro cliente, quest'ultimo deve essere visitato all'interno dello stesso percorso e prima del cliente i;
- per ogni cliente i, se D_i identifica un altro cliente, quest'ultimo deve essere visitato all'interno dello stesso percorso e dopo il cliente i.

Se per ogni cliente i, sia O_i sia D_i corrispondono al deposito, non è più necessario esplicitarli e durante la risoluzione del problema non è più necessario controllare che vengano rispettate le condizioni relative al partizionamento dei clienti negli h insiemi relativi agli h percorsi e all'ordine con cui i clienti vengono serviti all'interno del singolo percorso. Per riferirsi a quest'ultimo problema, Toth e Vigo in [1] utilizzano il nome VRP with Simultaneous Pick-up and Delivery (VRPSPD).

Il VRPPD è una generalizzazione del CVRP che può essere ottenuto dal primo nel caso in cui per ogni cliente i, O_i e D_i corrispondano al deposito e p_i valga 0; il VRPPD ed il VRPSPD sono quindi problemi di ottimizzazione NP-hard in senso forte.

Formulazione del VRPSPD

Il VRPSPD, così come il CVRP e il VRPPD, può essere descritto come un problema su grafo. $G = (V, A)$, dove $V = \{0..n\}$ è l'insieme dei vertici e A l'insieme degli archi. I vertici $1..n$ identificano i clienti mentre il vertice 0 identifica il deposito; ad ogni arco $(i, j) \in A$ è associato un costo non negativo c_{ij}, ossia il tempo necessario o la distanza da percorrere per andare da i a j.

Toth e Vigo in [1] descrivono tre differenti approcci base per modellare il VRP: le *vehicle flow formulations*, le *commodity flow formulations* e le *set-partitioning formulations*. I modelli del primo tipo utilizzano una variabile intera per ogni arco (spigolo) del grafo il cui valore rappresenta il numero delle volte che l'arco (spigolo) è percorso da un veicolo. Questi modelli sono particolarmente adatti nei casi in cui il costo della soluzione può essere espresso come la somma dei costi associati agli archi (spigoli) e quando i vincoli si limitano a considerare la sequenza con cui vengono visitati i clienti all'interno di un singolo ciclo (percorso); non riescono però a modellare varie questioni pratiche, ad es. il costo di una soluzione che dipenda dall'ordine globale con cui vengono visitati i clienti o dal tipo di veicolo assegnato ad un determinato ciclo.

Alla seconda classe appartengono i modelli in cui viene associata ad ogni arco (spigolo) un'ulteriore variabile intera che rappresenta il carico del veicolo che lo percorre.

I modelli dell'ultimo tipo sono caratterizzati da un numero esponenziale di variabili binarie, ognuna delle quali è associata ad un distinto percorso ammissibile; il VRP viene formulato come un Set-Partitioning Problem (SPP) la cui risoluzione consiste nel determinare l'insieme di percorsi di costo minimo che permettano di visitare i clienti una sola volta e che, possibilmente, soddisfino ulteriori vincoli. Il principale vantaggio di questo tipo di approccio è che permette di modellare le più svariate funzioni

obiettivo (ad es. quelle che non è possibile gestire con le *vehicle flow formulations*); esso richiede però l'utilizzo di un numero molto grande di variabili.

Domanda indivisibile

Nella versione base del VRPSPD la domanda di un cliente deve essere soddisfatta in un'unica visita. Il modello di programmazione misto-intera del VRPSPD simmetrico proposto è un'estensione della *commodity flow formulation* proposta in Garvin e altri[2]. N indica il numero di clienti suddivisi in due sottoinsiemi disgiunti P e D, rispettivamente l'insieme dei clienti caratterizzati da *pick-up demand* e l'insieme dei clienti caratterizzati da *delivery demand* (definendo $N_p = |P|$ e $N_d = |D|$ si ha che $N = N_p + N_d$); p_i è la *pick-up demand* del cliente i ($i \in P,\ p_i > 0$); d_i è *delivery demand* del cliente i ($i \in D,\ d_i < 0$). (c_{ij}) è la matrice dei costi; c_{ij} rappresenta la distanza da percorrere per andare da i a j, con $i, j \in \{0..N\}$; si assume che la matrice dei costi (c_{ij}) soddisfi la disuguaglianza triangolare e sia simmetrica. h indica il numero di veicoli identici e Q la capacità di ognuno di essi.

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} \cdot x_{ij} \tag{1.1}$$

$$\text{s.t.} \quad \sum_{i=0}^{N} x_{ij} = 1 \qquad \forall j = 1..N \tag{1.2}$$

$$\sum_{j=0}^{N} x_{ij} = 1 \qquad \forall i = 1..N \tag{1.3}$$

$$\sum_{j=0}^{N} x_{0j} = h \tag{1.4}$$

$$\sum_{k=0}^{N} y_{jk} - \sum_{i=0}^{N} y_{ij} = \begin{cases} p_j & j \in P \\ 0 & j \in D \\ -\sum_{i \in P} p_i & j = 0 \end{cases} \qquad \forall j = 0..N \tag{1.5}$$

$$\sum_{i=0}^{N} z_{ij} - \sum_{k=0}^{N} z_{jk} = \begin{cases} 0 & j \in P \\ -d_j & j \in D \\ \sum_{i \in D} d_i & j = 0 \end{cases} \qquad \forall j = 0..N \tag{1.6}$$

$$y_{ij} + z_{ij} \leq Q \cdot x_{ij} \qquad \forall i, j = 0..N \tag{1.7}$$

$$x_{ij} \in \{0, 1\}\ y_{ij},\ z_{ij} \geq 0 \qquad \forall i, j = 0..N \tag{1.8}$$

Il significato delle variabili è il seguente:

- x_{ij} è associata all'arco (i, j) e vale 1 se e solo se un veicolo transita da i a j;
- y_{ij} è associata all'arco (i, j) e il suo valore indica l'ammontare del carico raccolto che viene trasportato da i a j (è la parte di carico corrente accumulato soddisfacendo le *pick-up demands* nel percorso effettuato fino ad i);
- z_{ij} è associata all'arco (i, j) e il suo valore indica l'ammontare complessivo del carico da consegnare che viene trasportato da i a j (è la parte di carico corrente necessario a soddisfare le *delivery demands* nel percorso che viene effettuato dopo essere ripartiti da i).

L'obiettivo è quello di minimizzare la lunghezza totale dei percorsi. I vincoli (1.2) e (1.3) assicurano che ogni cliente venga visitato da un unico veicolo; il vincolo (1.4) impone che ogni veicolo durante il suo percorso visiti il deposito; infine i vincoli (1.5)-(1.7) impediscono la presenza di sottocicli isolati e assicurano che il carico corrente del veicolo lungo il percorso non ecceda la capacità Q del veicolo stesso.

Domanda divisibile

In Mosheiov[3] viene proposto un modello per il VRPSPD simmetrico assumendo che la domanda di un cliente sia divisibile, ossia possa essere soddisfatta in due o più visite.

In base a questa ipotesi, in Mosheiov[3] il problema viene riformulato considerando solo clienti (customers) con domanda pari a -1 o 1 (clienti a cui nel seguito della trattazione ci si riferirà con il termine 1-customers) ed a differenza di quanto accade nel modello (1.1)-(1.8), in cui in ogni luogo è localizzato un unico cliente, nel modello proposto da Mosheiov in ogni luogo possono essere localizzati più 1-customers.

Siano P' e D' rispettivamente gli insiemi degli 1-customers caratterizzati da *pick-up demand* e *delivery demand*; si definiscano $M_p = |P'|$, $M_d = |D'|$ e $M = M_p + M_d$ e sia N il numero dei luoghi in cui sono localizzati gli M 1-customers, il MIP model proposto da Mosheiov in [3] per il VRPSPD è:

$$\min \sum_{i=0}^{M} \sum_{j=0}^{M} c_{ij} \cdot x_{ij} \tag{1.9}$$

$$\text{s.t.} \sum_{i=0}^{M} x_{ij} = 1 \qquad \forall j = 1..M \tag{1.10}$$

$$\sum_{j=0}^{M} x_{ij} = 1 \qquad \forall i = 1..M \tag{1.11}$$

$$\sum_{j=0}^{M} x_{0j} = h \tag{1.12}$$

$$\sum_{k=0}^{M} y_{jk} - \sum_{i=0}^{M} y_{ij} = \begin{cases} 1 & j \in P' \\ 0 & j \in D' \\ -M_p & j = 0 \end{cases} \qquad \forall j = 0..M \tag{1.13}$$

$$\sum_{i=0}^{M} z_{ij} - \sum_{k=0}^{M} z_{jk} = \begin{cases} 0 & j \in P' \\ 1 & j \in D' \\ -M_d & j = 0 \end{cases} \qquad \forall j = 0..M \tag{1.14}$$

$$y_{ij} + z_{ij} \leq Q \cdot x_{ij} \qquad \forall i, j = 0..M \tag{1.15}$$

$$x_{ij} \in \{0, 1\}\ y_{ij},\ z_{ij} \geq 0 \qquad \forall i, j = 0..M \tag{1.16}$$

È da notare come, assumendo che la domanda sia divisibile, si limitano i casi reali a cui il modello può essere applicato ed inoltre il numero di variabili e il numero di vincoli che lo caratterizzano diventano così alti da non permettere il calcolo della soluzione esatta in tempo ragionevole, anche su piccole istanze del problema (cfr. Mosheiov[4]).

1.2 Panoramica dei metodi risolutivi

Il TSP with Simultaneous Pick-up and Delivery (TSPSPD) è stato finora risolto solo in maniera euristica attraverso algoritmi costruttivi [4] [9] [10] e più recentemente

attraverso il tabu search [12]. Anche la soluzione del VRPSPD è quindi ottenuta nei casi reali attraverso euristiche.

Laporte e Semet [5] forniscono una panoramica sulle *euristiche classiche* utilizzate per risolvere i VRPs e le suddividono in tre classi principali. La prima è quella delle *euristiche costruttive* che definiscono gradualmente una soluzione ammissibile tenendo in considerazione il costo della soluzione che si sta costruendo. La seconda classe è quella delle *euristiche a due fasi* che sono suddivise a loro volta in due sottoclassi: gli algoritmi *cluster-first, route-second* e gli algoritmi *route-first, cluster-second*. Nei primi i clienti vengono raggruppati in insiemi ammissibili e poi viene costruito per ogni insieme un percorso; negli altri viene prima costruito un ciclo che visita tutti i clienti ed in base ad esso vengono poi definiti i percorsi che dovranno effettuare i veicoli.

L'ultima classe è quella degli *algoritmi di ricerca locale* che cercano di migliorare una soluzione ammissibile eseguendo una sequenza di scambi di archi (spigoli) o di vertici (nodi) all'interno del singolo percorso o tra percorsi diversi.

Gendreau, Laporte e Potvin [6] descrivono invece sei *metaeuristiche* applicabili ai VRPs: *simulated annealing* (SA), *deterministic annealing* (DA), *tabu search* (TS), *genetic algorithms* (GA), *ant systems* (AS) e *neural networks* (NN). I primi tre algoritmi ricevono in input una soluzione ammissibile x_1, passano iterativamente dalla soluzione x_t alla soluzione x_{t+1} che appartiene all'intorno di x_t e terminano al verificarsi di determinate condizioni. Se $f(x)$ rappresenta il costo di x, allora $f(x_{t+1})$ non necessariamente è minore di $f(x_t)$; di conseguenza occorre prestare particolare attenzione alla possibilità che l'algoritmo continui a valutare le stesse soluzioni. I GA e gli AS costruiscono invece ad ogni passo un insieme di soluzioni: nei GA ogni popolazione (insieme di soluzioni) viene ottenuta dalla precedente scartando gli in-

dividui (soluzioni) peggiori e combinando tra loro quelli migliori; negli AS le nuove soluzioni vengono costruite sulla base delle informazioni raccolte considerando le soluzioni create nelle iterazioni precedenti. Infine le NN implementano un meccanismo di apprendimento che gradualmente modifica un insieme di pesi fino a raggiungere una soluzione ammissibile. Tutti questi algoritmi sono caratterizzati da una serie di parametri che devono essere tarati sperimentalmente in base allo specifico problema che deve essere risolto.

In generale le euristiche classiche effettuano una ricerca limitata all'interno dello spazio delle soluzioni ammissibili e producono soluzioni di buona qualità a fronte di bassi tempi di calcolo. Le metaeuristiche invece, accettando soluzioni intermedie peggiori o addirittura inammissibili, effettuano una ricerca approfondita nello spazio delle soluzioni e producono così soluzioni di qualità maggiore rispetto a quelle prodotte dalle euristiche classiche ma richiedono tempi di calcolo molto più elevati.

Capitolo 2

Euristiche a due fasi per il VRPSPD

2.1 Ammissibilità forte e debole

Si consideri la formulazione (1.1)-(1.8) descritta nel capitolo precedente; sia t un ciclo facente parte di una soluzione del VRPSPD e sia l la lista ordinata delle visite a clienti che lo definisce. Il ciclo t è ammissibile in senso forte se e solo se i vincoli (1.7) (i vincoli di capacità) sono rispettati lungo tutto il percorso; è debolmente ammissibile se esiste una lista ordinata l', ottenuta grazie a una permutazione delle visite in l, che definisce un ciclo t' fortemente ammissibile. A sua volta la soluzione del VRPSPD è debolmente ammissibile se contiene uno o più percorsi debolmente ammissibili.

2.2 Iterated Tour Partitioning

In Haimovic and Rinnooy Kan [7] è stato proposto un algoritmo denominato Iterated Tour Partitioning (ITP) per risolvere il CVRP successivamente modificato in Altinkemer and Gavish [8]. ITP è un algoritmo *route-first, cluster-second*: riceve in ingresso un ciclo hamiltoniano in cui non è incluso il deposito, calcolato a prescindere dalle domande dei clienti, e lo partiziona in segmenti; gli estremi di ciascuno dei segmenti ottenuti vengono collegati al deposito così da ottenere un insieme di cicli che costituisce la soluzione del CVRP. Ogni segmento viene terminato quando la

visita del successivo cliente nel ciclo iniziale causerebbe una violazione dei vincoli di capacità; la sequenza in cui i clienti compaiono in ogni segmento è quella in cui compaiono nel ciclo hamiltoniano.

A partire dall'idea che caratterizza il funzionamento di ITP, Righini [18] definisce quattro nuovi algoritmi tour partitioning per risolvere il VRPSPD con domanda dei clienti divisibile. Gli algoritmi, STRONGCONSEC (SC), STRONGSKIP (SS), WEAKCONSEC (WC) e WEAKSKIP (WS) , vengono definiti considerando due possibili criteri da utilizzare nel calcolo della soluzione a partire dal ciclo hamiltoniano iniziale:

1. la possibilità di definire o meno un ciclo debolmente ammissibile (WEAK / STRONG);

2. la possibilità di saltare o meno un cliente durante la definizione di un ciclo (SKIP / CONSEC).

I quattro algoritmi sono stati qui modificati per poter essere utilizzati anche nel caso in cui la domanda dei clienti sia indivisibile:

- in SC il segmento viene terminato quando la visita del successivo cliente causerebbe una violazione dei vincoli di capacità; grazie a questa condizione di terminazione vengono costruiti solo cicli fortemente ammissibili;

- in SS i clienti che causerebbero una violazione dei vincoli di capacità se visitati vengono saltati ed il segmento viene terminato quando non è più possibile visitare ulteriori clienti garantendo che venga mantenuta l'ammissiblità forte;

- in WC il segmento viene terminato quando la visita del successivo cliente farebbe aumentare l'ammontare del carico da consegnare o l'ammontare del

carico da raccogliere rendendolo superiore alla capacità del veicolo; in base a questa condizione di terminazione è possibile che il ciclo costruito sia debolmente ammissibile;

- in WS i clienti che se visitati farebbero aumentare l'ammontare del carico da consegnare o l'ammontare del carico da raccogliere rendendolo superiore alla capacità del veicolo, vengono saltati; il segmento viene terminato quando sia l'ammontare del carico da consegnare sia l'ammontare del carico da raccogliere sono pari alla capacità del veicolo o non possono più aumentare; è quindi possibile che il ciclo costruito sia debolmente ammissibile.

Se si considera il caso in cui la domanda è divisibile, gli algoritmi SC, SS e WC garantiscono che ogni veicolo visiti un numero di 1-customers compreso tra Q e $2Q$ mentre l'algoritmo WS garantisce che ogni veicolo ne visiti $2Q$, ossia garantisce che la capacità di ogni veicolo venga sfruttata al massimo e che quindi venga utilizzato il minor numero possibile di veicoli per servire tutti gli 1-customers.

L'operazione di partizionamento viene rieseguita N volte a partire da ciascun cliente e l'insieme di cicli di costo minimo tra tutti quelli ottenuti rappresenta la soluzione del VRPSPD. Se si assumesse che la domanda dei clienti sia divisibile si dovrebbero considerare tutti gli M 1-customers come possibili punti di partenza ma questo causerebbe un significativo aumento dei tempi di calcolo dal momento che M è generalmente molto più grande di N; ci si limita quindi a considerare come punti di partenza il primo degli 1-customers di ciascuno degli N luoghi che vengono raggiunti in base alla sequenza stabilita nel ciclo hamiltoniano iniziale.

Si osservi inoltre che il fatto che un cliente possa essere caratterizzato da *delivery demand* o da *pick-up demand* fa sì che il ciclo hamiltoniano iniziale sia asimmetrico

ai fini del partizionamento: se il ciclo iniziale viene invertito, gli algoritmi descritti possono produrre soluzioni differenti.

L'algoritmo di postprocessing utilizzato in WEAKCONSEC e WEAKSKIP per rendere le soluzioni prodotte fortemente ammissibili è quello proposto da Mosheiov in [3]: ogni ciclo debolmente ammissibile viene seguito dall'inizio alla fine ed i clienti che se serviti provocano inammissibilità vengono saltati. Una volta giunti alla fine del ciclo i clienti che non sono stati serviti vengono visitati in ordine inverso rispetto a quello in cui sono stati saltati. Questo metodo garantisce la definizione di cicli fortemente ammissibili; in particolare visitando tutti i clienti caratterizzati da *delivery demand* mentre si sta seguendo il ciclo debolmente ammissibile e servendo poi i clienti con *pick-up demand* in ordine inverso rispetto a quello in cui sono stati saltati si ottiene sempre l'ammissibilità forte.

Anche Mosheiov in [3] propone dei nuovi algoritmi tour partitioning per risolvere il VRPSPD definendoli a partire da ITP: Exhaustive ITP (EITP) e Full Capacity ITP (FCITP); i due algoritmi sono rispettivamente una versione primitiva degli algoritmi SC e WS definiti da Righini in [18].

I limiti principali di EITP e FCITP sono due:

1. il numero di volte per cui viene rieseguita l'operazione di partizionamento è limitato dal numero di 1-customers che vengono visitati nel primo segmento della prima soluzione ottenuta: così facendo non viene tenuto in considerazione l'"effetto-coda" dovuto alla presenza dell'ultimo ciclo, in cui la capacità del veicolo non viene sfruttata in maniera ottima;

2. non viene tenuta presente l'asimmetria insita nel ciclo hamiltoniano iniziale che quindi viene utilizzato come input un'unica volta, considerando un solo verso di percorrenza.

2.3 Errore di approssimazione

Il *worst-case error bound* rappresenta un limite superiore all'errore di approssimazione di un'euristica ed è indicato in percentuale rispetto al valore della soluzione ottima.

Relativamente ad un cliente, si denoti la distanza che lo separa dal deposito con il termine *distanza radiale*; nel ciclo compiuto da ogni veicolo è possibile distinguere la distanza coperta per raggiungere il primo cliente a partire dal deposito e per tornare al deposito a partire dall'ultimo cliente da quella percorsa a partire dal primo cliente fino a raggiungere l'ultimo cliente da visitare, ossia è possibile distinguere la distanza radiale percorsa da quella locale. Così come il singolo percorso, anche l'insieme di percorsi che rappresenta la soluzione del VRPSPD può essere caratterizzato dalla distanza radiale complessiva e dalla distanza locale complessiva percorse dai veicoli.

Mosheiov [3] determina *worst-case error bounds* per EITP e FCITP; per ognuno degli algoritmi egli ottiene un *upper bound* al valore della soluzione combinando tre diversi *upper bounds*:

- un *upper bound* sul numero dei veicoli;
- un *upper bound* sulla distanza radiale complessiva percorsa dai veicoli;
- un *upper bound* sulla distanza locale complessiva percorsa dai veicoli.

Il *worst-case error bound* è quindi ottenuto dividendo l'*upper bound* per il *lower bound* $max\{\frac{1}{Q}\bar{R}N, L(H)\}$ dove Q rappresenta la capacità dei veicoli, $\bar{R}$ il valore della distanza radiale media, N il numero dei clienti da visitare nel caso in cui la domanda dei clienti sia indivisibile (N è sostituito da M in caso contrario) e $L(H)$ la lunghezza del ciclo hamiltoniano iniziale.

Rifacendosi alla *worst-case analysis* effettuata da Mosheiov in [3] sono stati determinati gli *upper bounds* ai valori delle soluzioni trovate con i nuovi algoritmi definiti in questo capitolo sia nel caso in cui si assuma che la domanda dei clienti sia divisibile sia nel caso contrario; il *lower bound* sopraccitato è valido in entrambi i casi.

2.3.1 Domanda divisibile

In SC, SS e WC il minimo numero di 1-customers serviti con un veicolo è pari a Q ed il massimo numero di veicoli utilizzati è quindi $\lceil \frac{M}{Q} \rceil$. In WS invece, poiché possono essere definiti cicli debolmente ammissibili formati da visite a clienti che non appaiono in maniera consecutiva nel ciclo iniziale, ogni veicolo (tranne l'ultimo) serve Q 1-customers caratterizzati da *pick-up demand* e Q 1-customers caratterizzati da *delivery demand* ed il numero massimo di veicoli utilizzati è quindi $max\{\lceil \frac{M_p}{Q} \rceil, \lceil \frac{M_d}{Q} \rceil\}$.

In tutti e quattro gli algoritmi la distanza radiale percorsa da ogni veicolo è limitata da $2R_{max}$ dove R_{max} rappresenta la distanza radiale massima.

La distanza locale complessiva percorsa dai veicoli in SC è limitata dalla lunghezza del ciclo hamiltoniano $L(H)$, infatti i cicli fortemente ammissibili sono definiti da visite a clienti che compaiono in maniera consecutiva nel ciclo iniziale. In SS, poiché la lunghezza di ogni ciclo fortemente ammissibile è limitata da $L(H)$ e ciascuno di essi deve essere attraversato al più una volta, è pari a $\lceil \frac{M}{Q} \rceil L(H)$, mentre in WC, poiché ogni ciclo è definito da visite a clienti che compiono in maniera consecutiva nel ciclo iniziale e deve essere attraversato al più due volte, esso è pari a $2L(H)$. In WS infine l'*upper bound* della distanza locale percorsa da ogni veicolo è $2L(H)$, ogni ciclo debolmente ammissibile deve essere infatti attraversato al più due volte per generare un ciclo fortemente ammissibile riordinando la sequenza delle visite ai clienti.

Gli *upper bounds* delle soluzioni trovate da SC, SS, WC e WS sono quindi:

- $\lceil \frac{M}{Q} \rceil 2R_{max} + L(H)$;
- $\lceil \frac{M}{Q} \rceil \left[2R_{max} + L(H)\right]$;
- $\lceil \frac{M}{Q} \rceil 2R_{max} + 2L(H)$;
- $max \left\{ \lceil \frac{M_p}{Q} \rceil, \lceil \frac{M_d}{Q} \rceil \right\} \left[2R_{max} + 2L(H)\right]$.

2.3.2 Domanda indivisibile

L'analisi descritta nella sottosezione precedente risulta essere ancora valida tranne per quanto riguarda la determinazione dell'*upper bound* del numero dei veicoli che è $\lceil \frac{N}{2} \rceil$ per tutti gli algoritmi.

Gli *upper bounds* delle soluzioni trovate da SC, SS, WC e WS sono rispettivamente:

- $\lceil \frac{N}{2} \rceil 2R_{max} + L(H)$;
- $\lceil \frac{N}{2} \rceil \left[2R_{max} + L(H)\right]$;
- $\lceil \frac{N}{2} \rceil 2R_{max} + 2L(H)$;
- $\lceil \frac{N}{2} \rceil \left[2R_{max} + 2L(H)\right]$.

I *worst-case error bounds* determinati sia nel caso in cui la domanda dei clienti sia divisibile sia nel caso contrario sono interessanti solo dal punto di vista teorico; infatti dipendono dai dati delle istanze considerate. Maggior chiarezza sul comportamento dei quattro algoritmi può essere ottenuta a partire dall'osservazione fatta da Mosheiov in [3]: la differenza tra i due modi in cui FCITP e EITP definiscono la soluzione riflette il trade-off tra l'obiettivo di minimizzare la distanza radiale complessivamente percorsa utilizzando il minor numero possibile di veicoli (obiettivo (1))e quello di minimizzare la distanza locale complessivamente percorsa rispettando l'ordine di visita dei

clienti stabilito dal ciclo hamiltoniano iniziale (obiettivo (2)). Questa osservazione è valida anche per WS e SC. Per quanto riguarda invece SS e WC risulta essere vera la seguente: rispetto a SC, SS considera l'obiettivo (1) e dà peso minore al (2); WC, rispetto a WS, dà peso minore all'obiettivo (1) e maggiore al (2).

In base all'osservazione fatta, Mosheiov ipotizza che FCITP funzioni meglio di EITP quando i clienti sono vicini gli uni agli altri e la distanza radiale media è significativamente superiore alla distanza media tra due clienti, e viceversa EITP sia da preferirsi quando i clienti sono localizzati vicino al deposito e la distanza media tra due clienti è significativamente superiore alla distanza radiale media; questa ipotesi viene confermata dagli esperimenti descritti in [3]. Tuttavia quale tra EITP e FCITP sia in definitiva il migliore Mosheiov non è stato in grado di stabilirlo e questo, visto che EITP e FCITP sono le versioni primitive di SC e WS, spinge ad una valutazione e ad un confronto sperimentale delle prestazioni di SC, SS, WC e WS.

2.4 Risultati sperimentali

Gli esperimenti condotti sui quattro algoritmi tour partitioning presentati nella sezione precedente possono essere suddivisi in due classi principali. Lo scopo degli esperimenti della prima classe è stato quello di valutare:

1. come le prestazioni dei quattro algoritmi vengono influenzate dal considerare il ciclo hamiltoniano iniziale in entrambi i sensi di percorrenza;

2. la robustezza dei quattro algoritmi rispetto alla qualità del ciclo hamiltoniano iniziale;

3. la qualità delle soluzioni calcolate dai quattro algoritmi;

sia nel caso in cui si assuma che la domanda di un cliente sia divisibile sia nel caso contrario.

Lo scopo della seconda classe di esperimenti è stato invece quello di valutare quanto incide sulla qualità delle migliori soluzioni calcolabili con gli algoritmi tour partitioning assumere o meno che la domanda sia divisibile.

Negli esperimenti svolti il ciclo hamiltoniano iniziale viene calcolato attraverso quattro differenti euristiche per il TSP: Nearest Insertion [11], Cheapest Insertion [11] (entrambi inizializzati con la coppia di clienti più vicini tra loro), Farthest Insertion [11] e Largest Insertion [12] (entrambi inizializzati con la coppia di clienti più lontani tra loro).

Per i test sono stati utilizzati istanze euclidee generate in maniera casuale con la stessa tecnica descritta in [3]: 6 insiemi di 50 istanze, caratterizzate da un numero di clienti pari a 50 (insiemi 1, 2, 3) e 100 (insiemi 4, 5, 6) uniformemente distribuiti in un quadrato di lato 2 centrato rispetto all'origine del sistema cartesiano nella quale è localizzato il deposito e con domanda uniformemente distribuita negli intervalli $[-10, 10]$ (insiemi 1, 4), $[-20, 20]$ (insiemi 2, 5) e $[-40, 40]$ (insiemi 3, 6). Questo corrisponde ad avere un numero medio di 250, 500, 1000, 500, 1000, e 2000 1-customers nel caso in cui si assuma che la domanda sia divisibile.

Nella prima classe di esperimenti, nel caso in cui la domanda sia divisibile la capacità dei veicoli è pari a 10 indipendentemente dall'insieme considerato; se si assume invece che la domanda sia indivisibile la capacità dei veicoli è pari al minimo indispensabile per servire un cliente: 10 (insiemi 1, 4), 20 (insiemi 2, 5) e 40 (insiemi 3, 6). Negli esperimenti appartenenti alla seconda classe, invece, la capacità dei veicoli viene fatta variare.

Verso di percorrenza del ciclo iniziale

Per valutare come l'inversione del ciclo iniziale influisca sulle prestazioni dei quattro algoritmi, essi sono stati testati su tutte e 300 le istanze utilizzando 4 cicli iniziali prodotti con le euristiche sopraccitate. I risultati sono riportati nelle tabelle 2.1 e 2.2: essi indicano la differenza massima e quella media tra le migliori soluzioni trovate invertendo il ciclo iniziale; gli errori sono indicati in percentuale rispetto ai migliori valori trovati ed i valori medi sono calcolati sulle 50 istanze di ogni insieme e sui 4 cicli iniziali.

Sia nel caso in cui la domanda sia divisibile sia nel caso contrario, le prestazioni degli algoritmi che richiedono ammissibilità forte sono molto sensibili all'inversione del ciclo iniziale e lo stesso vale anche per gli algoritmi che saltano clienti. In particolare la lunghezza dei cicli costruiti da WEAKSKIP e STRONGSKIP dipende fortemente dall'ordine in cui i clienti caratterizzati da *delivery demand* e quelli caratterizzati da *pick-up demand* compaiono nel ciclo hamiltoniano iniziale più che dalla loro localizzazione: clienti molto lontani gli uni dagli altri possono essere inseriti nello stesso ciclo se il ciclo iniziale viene percorso in un senso o possono appartenere a due cicli distinti se viene considerato l'altro senso di percorrenza. L'algoritmo WEAKCONSEC è quello che risulta essere meno influenzato poiché definisce ciclo debolmente ammissibili in cui vengono visitati clienti che compaiono consecutivamente nel ciclo iniziale.

In generale il funzionamento di tutti e quattro gli algoritmi tour partitioning è significativamente influenzato dal senso di percorrenza del ciclo iniziale.

Qualità del ciclo iniziale

Per valutare la relazione tra la qualità del ciclo hamiltoniano iniziale e quella della soluzione del VRPSPD sono stati confrontati, per ogni insieme di problemi, la lunghezza media dei cicli iniziali e la lunghezza media delle corrispondenti soluzioni ottenute dagli algoritmi tour partitioning; il confronto è stato effettuato per ogni euristica utilizzata per risolvere il TSP e per ognuno dei quattro algoritmi sotto esame considerando sia domanda divisibile sia indivisibile.

Come è possibile osservare dai risultati riportati nelle tabelle 2.3 e 2.4, in entrambi i casi la qualità delle soluzioni calcolate dagli algoritmi tour partitioning va di pari passo con la qualità dei cicli iniziali: per ogni algoritmo e per ogni insieme di problemi, le migliori soluzioni ottenute sono quelle relative a cicli iniziali prodotti con Farthest Insertion e Largest Insertion che, fra quelle utilizzate, sono le euristiche più efficaci per risolvere il TSP. È da notare come le soluzioni calcolate a partire dai cicli ottenuti con Largest Insertion (vedi valori in grassetto) a volte dominano quelle calcolate a partire dai cicli prodotti con Farthest Insertion; in media sono quest'ultime ad essere le migliori.

Qualità delle soluzioni

Si consideri il caso in cui la domanda di un cliente sia divisibile, i risultati della tabella 2.3 mostrano come per ogni insieme di problemi i quattro algoritmi tour partitioning possono essere classificati in base alle loro prestazioni: WEAKSKIP risulta essere il migliore, seguito da STRONGSKIP, WEAKCONSEC e STRONGCONSEC. Dal momento che WEAKSKIP e STRONGCONSEC sono le varianti degli algoritmi FCITP e EITP ideati da Mosheiov che non era riuscito con i suoi esperimenti a stabilire quale dei due fosse il migliore, è possibile affermare come le modifiche

proposte abbiano permesso di superare questo limite: le prestazioni di WEAKSKIP sono migliorate molto rispetto a FCITP ed ora WEAKSKIP è nettamente migliore di STRONGCONSEC anche su piccole istanze del problema (insiemi 1 e 4) per le quali secondo Mosheiov EITP funzionava meglio di FCITP.

Nel caso in cui la domanda sia indivisibile è invece STRONGSKIP ad essere l'algoritmo tour partitioning migliore, seguito da WEAKSKIP, WEAKCONSEC e STRONGCONSEC. WEAKSKIP cerca di utilizzare la capacità dei veicoli nella maniera ottimale rinunciando alla definizione di cicli fortemente ammissibili; essendo però la domanda indivisibile, i cicli che determina possono essere formati da clienti che nel ciclo hamiltoniano iniziale compaiono molto distanti gli uni dagli altri: è questo a far sì che STRONGSKIP garantisca prestazioni migliori.

Oltre che dalla lunghezza totale dei cicli, la soluzione del VRPSPD è anche caratterizzata dal numero di veicoli utilizzati. Considerando il costo derivante dall'utilizzo di un veicolo (composto dal costo dei veicoli e dai salari dei guidatori) e quello derivante dal percorrere una distanza maggiore rispetto alla minima si intuisce come la minimizzazione del numero dei veicoli utilizzati sia spesso l'obiettivo più importante, perciò una corretta valutazione degli algoritmi che risolvono i VRPs deve tenere in considerazione anche questo aspetto. Le tabelle 2.5 e 2.6 mostrano rispettivamente nel caso in cui la domanda sia divisibile e nel caso contrario il numero medio di veicoli richiesto da ogni algoritmo tour partitioning relativamente a ciascun insieme di problemi; ogni valore medio è stato calcolato sulle 50 istanze e sui 4 cicli iniziali per ogni istanza.

In entrambi i casi WEAKSKIP risulta essere l'algoritmo migliore ed in particolare nel caso in cui la domanda sia divisibile permette di trovare il numero ottimo di veicoli da utilizzare poiché sfrutta interamente la capacità di ciascuno di essi. STRONGSKIP

trova soluzioni con un numero di veicoli molto vicino al minimo valore trovato con WEAKSKIP, il valore del gap è al massimo pari al 3% rispetto al valore minimo nel caso in cui la domanda sia divisibile e al 4% nel caso contrario. Al contrario STRONGCONSEC e WEAKCONSEC suddividono il ciclo hamiltoniano iniziale in un numero molto più alto di cicli: il numero di veicoli che caratterizza le soluzioni trovate da questi due algoritmi è maggiore rispetto al valore minimo di una percentuale compresa tra il 25% e il 70% nel caso in cui la domanda sia divisibile e compresa tra il 35% e il 65% nell'altro caso.

Alla luce dei risultati fin qui ottenuti è possibile stabilire come il miglior algoritmo tour partitioning da utilizzarsi nel caso in cui sia valida l'assunzione di domanda divisibile sia WEAKSKIP mentre nel caso contrario sia STRONGSKIP. Questi due algoritmi sono quindi quelli che vengono impiegati nella seconda classe di esperimenti.

Domanda divisibile vs. domanda indivisibile

Quello che si è cercato di capire grazie alla seconda classe di esperimenti è l'effetto che genera l'introduzione del vincolo che la domanda dei clienti sia indivisibile sui migliori *upper bounds* calcolabili con gli algoritmi tour partitioning sotto esame. A tal fine sono stati eseguiti una serie di esperimenti facendo variare la capacità Q dei veicoli da un minimo di 10 ad un massimo di 100 ed utilizzando l'algoritmo WEAKSKIP nel caso di domanda divisibile e STRONGSKIP nel caso contrario.

Le tabelle 2.7 e 2.8 mostrano rispettivamente la lunghezza media delle soluzioni calcolate dai due algoritmi ed il numero medio di veicoli che le caratterizza per ciascun insieme di problemi al variare di Q; i valori migliori per ogni singolo caso, identificato dal particolare insieme e dal valore di Q, sono evidenziati in grassetto.

Far aumentare la capacità dei veicoli ha l'effetto di vincolare meno il problema,

dai risultati riportati nella tabella 2.7 si può infatti notare come le soluzioni calcolate da STRONGSKIP e da WEAKSKIP migliorino a mano a mano che Q aumenta.

Si consideri ora il il gap tra la lunghezza media delle soluzioni calcolate da STRONGSKIP e da WEAKSKIP: poiché l'aumento di Q compensa in parte la presenza del vincolo di domanda indivisibile, più il gap tra le soluzioni è piccolo con valori di capacità bassi, meno il vincolo incide sulla bontà delle migliori soluzioni calcolabili con gli algoritmi tour partitioning. Da un esame dei risultati presenti in tabella 2.7 emerge che il gap non solo diminuisce abbastanza rapidamente all'aumentare di Q in ogni insieme, ma per alcuni di essi (insiemi 1,2,4,5) esiste un valore di Q tra quelli presi in considerazione (rispettivamente 30,50,40,100) oltre il quale le soluzioni calcolate con STRONGSKIP sono migliori (fino al 6% sul valore minimo) di quelle calcolate con WEAKSKIP. Da un più attento esame si evince inoltre che l'effetto sulla lunghezza media delle migliori soluzioni calcolabili con gli algoritmi tour partitioning generato dal vincolo di domanda indivisibile cessa di esistere quando Q è almeno pari a circa $1/8$ del numero medio di 1-customers che caratterizzano l'insieme.

Relativamente al numero medio di veicoli, invece, dall'esame della tabella 2.8 emerge che, per tutti gli insiemi tranne il 6, esiste un valore di Q tra quelli presi in considerazione (rispettivamente 20, 50, 100, 30, 70), per il quale le soluzioni calcolate con STRONGSKIP, rispetto a quelle calcolate con WEAKSKIP, consentono di utilizzare in media circa lo stesso numero di veicoli. Oltre tale valore le soluzioni calcolate con STRONGSKIP impongono al limite, in media, di utilizzare circa un solo veicolo in più. Il vincolo di domanda indivisibile cessa di avere effetti significativi sul numero medio di veicoli quando Q è almeno pari a circa $1/10$ del numero medio di 1-customers.

In definitiva sono due i risultati fondamentali emersi dalla seconda classe di

esperimenti:

- il vincolo di indivisibilità della domanda incide sulla bontà delle migliori soluzioni ottenibili con gli algoritmi tour partitioning in maniera crescente col numero di clienti che devono essere serviti e col valore assoluto medio della loro domanda;

- esiste un valore minimo di Q, pari a circa $1/8$ del numero di 1-customers, al di sopra del quale assumere che la domanda dei clienti sia indivisibile permette di calcolare con i quattro algoritmi tour partitioning sotto esame (in particolare con STRONGSKIP) soluzioni migliori rispetto a quelle che sarebbero calcolate con domanda divisibile.

	STRONGCONSEC		WEAKCONSEC		STRONGSKIP		WEAKSKIP	
Set	max(%)	av.(%)	max(%)	av.(%)	max(%)	av.(%)	max(%)	av.(%)
1	12,00	3,38	7,43	2,02	13,81	3,66	10,17	2,73
2	9,93	2,22	3,00	0,80	12,32	3,36	8,51	1,97
3	4,26	1,10	1,60	0,37	15,57	3,75	10,70	1,60
4	7,67	2,52	4,11	0,96	10,23	2,44	7,19	1,69
5	5,20	1,55	1,77	0,46	11,12	2,51	5,96	1,42
6	2,79	0,86	1,06	0,21	13,57	2,40	6,05	1,39

Tabella 2.1: domanda divisibile, influenza dell'inversione del ciclo hamiltoniano iniziale

	STRONGCONSEC		WEAKCONSEC		STRONGSKIP		WEAKSKIP	
Set	max(%)	av.(%)	max(%)	av.(%)	max(%)	av.(%)	max(%)	av.(%)
1	11,98	2,94	4,05	1,17	14,40	4,39	16,65	3,53
2	14,71	3,40	5,23	1,18	18,99	4,28	12,91	3,54
3	14,96	3,21	6,14	1,29	19,47	4,75	15,21	3,64
4	8,92	2,41	2,63	0,63	15,01	3,73	10,69	2,66
5	10,96	2,28	2,39	0,69	14,81	3,95	14,74	3,14
6	11,30	2,39	2,43	0,75	12,89	3,29	8,39	2,95

Tabella 2.2: domanda indivisibile, influenza de l'inversione del ciclo hamiltoniano iniziale

Set	TSP alg.	TSP	SC	WC	SS	WS
1	FI	6,1247	39,0889	36,1189	33,9053	33,4204
1	LI	6,1905	38,8893	36,0474	34,1436	33,4391
1	CI	6,7725	39,4349	36,8498	34,9567	34,4964
1	NI	6,9467	39,0230	36,5193	34,9081	34,6451
2	FI	6,0654	76,1063	70,1401	58,6639	57,3602
2	LI	6,1513	76,2247	70,7125	58,8304	57,3086
2	CI	6,6466	76,4621	70,8620	60,0188	58,6940
2	NI	6,8388	76,3412	71,0239	60,1574	59,0606
3	FI	6,0887	153,3877	145,8903	109,9430	105,1569
3	LI	6,1723	152,4870	144,8925	109,8336	105,0915
3	CI	6,6585	153,4625	146,1263	111,3712	106,7722
3	NI	6,8834	153,7882	146,3285	111,8894	107,4841
4	FI	8,4476	73,3392	67,4494	60,6202	59,6457
4	LI	8,6351	73,7133	67,4472	60,8401	59,8954
4	CI	9,2785	74,5878	68,5689	61,9609	61,1077
4	NI	9,5027	74,9947	68,9417	62,3812	61,4016
5	FI	8,4632	149,0551	136,2708	111,1761	106,9689
5	LI	8,5833	148,7665	136,4488	111,0553	107,0815
5	CI	9,2876	149,4502	137,2429	113,0045	109,0738
5	NI	9,6144	149,3323	137,6564	113,8243	109,8268
6	FI	8,4336	302,1439	286,0732	209,1443	199,8334
6	LI	8,5471	301,9013	285,0525	209,2434	200,0801
6	CI	9,2695	301,8966	286,1481	212,2864	203,4673
6	NI	9,5396	302,5810	286,0114	213,3184	204,1219

Tabella 2.3: domanda divisibile, costi

Capitolo 3

Algoritmi di ricerca locale

3.1 Considerazioni e definizioni di base

Durante la definizione di algoritmi di ricerca locale per il VRPSPD è importante considerare il fatto che nei cicli che formano la soluzione è molto probabile che i clienti caratterizzati da *delivery demand* siano serviti all'inizio mentre i clienti caratterizzati da *pick-up demand* alla fine; di conseguenza se si inverte un ciclo è molto probabile che i vincoli di capacità siano violati. È quindi consigliabile utilizzare definizioni di intorni simili a quelli efficacemente utilizzati nella versione asimmetrica del CVRP che non si basano sull'inversione di parte della soluzione. Tre intorni identificati con i nomi RELOCATE, EXCHANGE e CROSS utilizzati per risolvere il CVRP asimmetrico con algoritmi di ricerca locale sono illustrati in [13] e [14] e possono essere adattati per risolvere il VRPSPD.

Nella definizione dei nuovi intorni è da tener infine presente che accettare soluzioni debolmente ammissibili è un modo intelligente di esplorare vasti intorni rimanendo vicini alla regione ammissibile.

Durante la ricerca locale una soluzione accettabile è rappresentata da un insieme di h cicli fortemente ammissibili; ogni ciclo t è rappresentato da una lista ordinata di C_t visite ai clienti, identificate da $v_{t,k}$ con $k = 1, ..., C_t$; ogni visita $v_{t,k} = (i, r_i)$

è rappresentata dall'indice i del cliente che assume valori tra 1 e N e un ammontare r_i del carico da raccogliere ($r_i > 0$) o da consegnare ($r_i < 0$) dal o al cliente i. Al fine di controllare l'ammissibilità debole l'ammontare totale del carico raccolto e del carico da consegnare, p_t e d_t, sono memorizzati per ogni veicolo t. Al fine di controllare l'ammissibilità forte le seguenti informazioni addizionali sono associate ad ogni visita $v_{t,k}$:

- l'ammontare totale del carico raccolto che si trova a bordo del veicolo t subito prima e subito dopo la visita, indicati rispettivamente con $p^{<}_{t,k}$ e $p^{>}_{t,k}$;
- l'ammontare totale del carico da distribuire che si trova a bordo del veicolo t subito prima e subito dopo la visita, indicati rispettivamente con $d^{<}_{t,k}$ e $d^{>}_{t,k}$;
- il valore massimo del carico totale che il veicolo t trasporta prima e dopo la visita, indicati rispettivamente con $L^{<}_{t,k} = max_{j \leq k} \left\{ p^{<}_{t,j} + d^{<}_{t,j} \right\}$ e $L^{>}_{t,k} = max_{j \geq k} \left\{ p^{>}_{t,j} + d^{>}_{t,j} \right\}$.

Valgono le seguenti relazioni:

$$
\begin{array}{lll}
p^{<}_{t,k} + max\left\{r_i, 0\right\} = p^{>}_{t,k} & \forall t = 1, ..., h, \ \forall k = 1, ..., C_t \\
d^{<}_{t,k} - max\left\{-r_i, 0\right\} = d^{>}_{t,k} & \forall t = 1, ..., h, \ \forall k = 1, ..., C_t \\
p^{>}_{t,k} = p^{<}_{t,k+1} & \forall t = 1, ..., h, \ \forall k = 1, ..., C_t - 1 \\
d^{>}_{t,k} = d^{<}_{t,k+1} & \forall t = 1, ..., h, \ \forall k = 1, ..., C_t - 1 \\
p^{<}_{t,1} = 0 & \forall t = 1, ..., h \\
p^{>}_{t,C_t} = p_t & \forall t = 1, ..., h \\
d^{<}_{t,1} = d_t & \forall t = 1, ..., h \\
d^{>}_{t,C_t} = 0 & \forall t = 1, ..., h \\
L^{<}_{t,k} = max\left\{L^{<}_{t,k-1}, p^{<}_{t,k} + d^{<}_{t,k}\right\} & \forall t = 1, ..., h, \ \forall k = 2, ..., C_t \\
L^{>}_{t,k} = max\left\{p^{>}_{t,k} + d^{>}_{t,k}, L^{>}_{t,k+1}\right\} & \forall t = 1, ..., h, \ \forall k = 1, ..., C_t - 1 \\
L^{<}_{t,1} = d_t & \forall t = 1, ..., h \\
L^{>}_{t,C_t} = p_t & \forall t = 1, ..., h
\end{array}
$$

3.2 Intorni

Gli algoritmi di ricerca locale che si basano sui nuovi intorni vengono inizializzati con una soluzione calcolata dalle euristiche a due fasi esaminate nel capitolo precedente; esse possono generare visite multiple nel caso in cui la domanda dei clienti sia divisibile. In tal caso, è importante notare che la ricerca locale non può né raggruppare né suddividere le visite ad un cliente.

Nel resto della sezione S indica il numero delle visite ai clienti nella soluzione con cui l'algoritmo di ricerca locale viene inizializzato: il suo valore è pari al numero dei clienti nel caso in cui la domanda sia indivisibile ed è compreso tra N (il numero dei luoghi) ed M (il numero degli 1-customers) nel caso contrario.

Per ogni nuovo intorno che viene di seguito definito vengono esaminate le procedure per generare nuove soluzioni, per controllare la loro ammissibilità, per valutarle e per aggiornare la soluzione corrente.

3.2.1 RELOCATE

Questo intorno include tutte le soluzioni fortemente ammissibili ottenute cancellando la visita a un cliente $v_{t_1,k_1} = (i, r_i)$ dal ciclo t_1 a cui appartiene e reinserendola in un ciclo t_2.

Generazione. Una nuova soluzione è generata per ogni visita e per ogni posizione di inserimento ammissibile; è possibile che $t_1 = t_2$ e cioè che la visita sia reinserita nello stesso ciclo in una posizione più conveniente. Il numero delle soluzioni considerate è $O(S^2)$. Per risparmiare tempo di calcolo il ciclo t_2 deve essere scandito dall'inizio se $r_i < 0$ e dalla fine se $r_i > 0$; la scansione va terminata non appena viene raggiunta la prima posizione di inserimento inammissibile.

Controllo dell'ammissibilità. È effettuato in tempo costante. Devono essere verificate le seguenti condizioni perché la visita (i, r_i) possa essere inserita tra v_{t_2,k_2} e $v_{t_2,k_{2+1}}$:

$$L^{>}_{t_2,k_2} + max\left\{r_i, 0\right\} \leq Q$$

$$L^{<}_{t_2,k_{2+1}} + max\left\{-r_i, 0\right\} \leq Q$$

Valutazione. È effettuata in tempo costante: due archi sono sostituiti da uno quando la visita è eliminata dal ciclo a cui appartiene ed un arco è sostituito da due quando la visita è inserita nella nuova posizione; la differenza tra la lunghezza degli archi inseriti e quelli cancellati rappresenta il costo del passo di ricerca locale.
Aggiornamento. I valori associati ai cicli t_1 e t_2 devono essere parzialmente ricalcolati, la complessità dell'operazione in termini di tempo è quindi $O(Q)$.

3.2.2 EXCHANGE

In base alla definizione dell'intorno data da Vigo in [14], una nuova soluzione per il CVRP asimmetrico è ottenuta da quella corrente selezionando due clienti e scambiandoli fra di loro. Nel VRPSPD lo scambio di due visite può produrre soluzioni debolmente o fortemente ammissibili, di conseguenza sono stati definiti due intorni simili: EXCHANGESTRONG e EXCHANGEWEAK. Il primo contiene solo le soluzioni fortemente ammissibili ottenibili con uno scambio, il secondo include anche le soluzioni debolmente ammissibili.

EXCHANGESTRONG

Generazione. Sono considerate tutte le coppie ordinate di visite; lo scambio è ammesso anche se le visite appartengono allo stesso ciclo. Il numero delle soluzioni considerate è quindi pari a $O(S^2)$.

Controllo dell'ammissibilità. Se $t_1 \neq t_2$, perché la visita $v_{t_2,k_2} = (j, r_j)$ possa sostituire la visita $v_{t_1,k_1} = (i, r_i)$, devono essere verificate le seguenti condizioni relativamente a t_1:

$$L^{>}_{t_1,k_1} + max\{r_j, 0\} - max\{r_i, 0\} \leq Q$$

$$L^{<}_{t_1,k_1} + max\{-r_j, 0\} - max\{-r_i, 0\} \leq Q$$

e analoghe condizioni devono essere soddisfatte per t_2; l'operazione richiede quindi tempo costante. Se invece $t_1 = t_2$ (supponendo senza perdita di generalità che $k_1 < k_2$) il controllo è effettuato verificando che sia rispettato il vincolo di capacità su ogni arco compreso nel segmento del ciclo delimitato dalle visite v_{t_1,k_1} e v_{t_2,k_2}. In quest'ultimo caso la complessità dell'operazione in termini di tempo è $O(Q)$.

Valutazione. Il valore delle nuove soluzioni dipende solo dagli archi coinvolti nello scambio e può essere quindi calcolato in tempo costante; gli archi sono in generale quattro fatta eccezione per il caso in cui $t_1 = t_2$ e $k_2 = k_1 + 1$ oppure $t_1 = t_2$ e $k_1 = k_2 + 1$ (visite consecutive lungo lo stesso ciclo).

Aggiornamento. I valori delle strutture dati associate ai cicli t_1 e t_2 devono essere parzialmente ricalcolati, la complessità dell'operazione in termini di tempo è quindi $O(Q)$.

EXCHANGEWEAK

Generazione. Come in EXCHANGESTRONG sono considerate tutte le coppie ordinate di visite; il numero delle soluzioni considerate è quindi pari a $O(S^2)$.

Controllo dell'ammissibilità. Un ciclo è debolmente ammissibile se e solo se sia l'ammontare del carico da raccogliere sia l'ammontare del carico da consegnare non eccedono la capacità Q dei veicoli. Quindi se $t_1 \neq t_2$, perché la visita $v_{t_2,k_2} = (j, r_j)$ possa sostituire la visita $v_{t_1,k_1} = (i, r_i)$, devono essere verificate le seguenti condizioni

relativamente a t_1:

$$p_{t_1} - max\{r_i, 0\} + max\{r_j, 0\} \leq Q$$

$$d_{t_1} - max\{-r_i, 0\} + max\{-r_j, 0\} \leq Q$$

e analoghe condizioni devono essere soddisfatte per t_2; l'operazione richiede quindi tempo costante. Se invece $t_1 = t_2$ lo scambio è sempre possibile.

Valutazione. Una volta selezionate le due visite da scambiare, vengono anche scelte le migliori posizioni di inserimento ammissibili nei nuovi cicli. Si consideri il ciclo t_2 in cui la visita (j, r_j) è sostituita dalla visita (i, r_i): per risparmiare tempo di calcolo nella ricerca della posizione di inserimento si scandisce il ciclo dall'inizio se $r_i < 0$ e dalla fine se $r_i > 0$, e si termina non appena l'inserimento della visita provocherebbe inammissibilità. Se $t_1 \neq t_2$ la complessità dell'operazione in termini di tempo è $O(Q)$; se invece $t_1 = t_2$, è necessario considerare tutte le coppie di inserimenti possibili e la complessità diventa $O(Q^2)$.

Aggiornamento. L'operazione ha una complessità in termini di tempo pari a $O(Q)$ come nel caso di EXCHANGESTRONG.

3.2.3 CROSS

Questo intorno contiene tutte le soluzioni fortemente ammissibili ottenibili dalla sostituzione di due archi con altri due. Si prendono in considerazione due differenti cicli (t_1 e t_2) e li si suddivide in due parti distinte eliminando un arco da ciascuno di essi; si ricombina poi la prima parte di t_1 con la seconda di t_2 e la prima parte di t_2 con la seconda di t_1 ottenendo due nuovi cicli.

Generazione. Si esamina una nuova soluzione per ogni coppia ordinata di archi (a_1, a_2) dove a_1 collega $v_{t_1,k_1} = (i_1, r_{i_1})$ a $v_{t_1,k_1+1} = (j_1, r_{j_1})$ e a_2 collega $v_{t_2,k_2} =$

(i_2, r_{i_2}) a $v_{t_2,k_2+1} = (j_2, r_{j_2})$ con $t_1 \neq t_2$. Il numero delle soluzioni considerate è quindi ancora pari a $O(S^2)$.

Controllo dell'ammissibilità. L'operazione è eseguita in tempo costante: perché la coppia di archi (a_1, a_2) possa essere sostituita da quelli nuovi devono essere verificate le seguenti condizioni:

$$L^{<}_{t_1,k_1+1} - d^{<}_{t_1,k_1+1} + d^{<}_{t_2,k_2+1} \leq Q$$

$$L^{>}_{t_2,k_2} - p^{>}_{t_2,k_2} + p^{>}_{t_1,k_1} \leq Q$$

$$L^{<}_{t_2,k_2+1} - d^{<}_{t_2,k_2+1} + d^{<}_{t_1,k_1+1} \leq Q$$

$$L^{>}_{t_1,k_1} - p^{>}_{t_1,k_1} + p^{>}_{t_2,k_2} \leq Q$$

Valutazione. Il valore della nuova soluzione dipende solo dalla lunghezza dei quattro archi coinvolti nello scambio e può essere quindi calcolata in tempo costante.

Aggiornamento. Anche in questo intorno i valori associati ai cicli t_1 e t_2 devono essere parzialmente ricalcolati; la complessità dell'operazione in termini di tempo è quindi ancora pari a $O(Q)$.

3.2.4 3-ARC-EXCHANGE

Questo intorno include tutte le soluzioni fortemente ammissibili ottenibili dalla sostituzione di tre archi appartenenti a tre diversi cicli, effettuata in modo analogo a quello previsto in CROSS.

Generazione. Il numero delle soluzioni considerate è quindi ancora pari a $O(S^3)$; in particolare è da tener presente il fatto che per ogni tripla di archi eliminati esistono due differenti modi di ricombinare le parti in cui i tre cicli vengono suddivisi.

Controllo dell'ammissibilità. Valutazione. Aggiornamento. È valida l'analisi presentata in CROSS per le corrispondenti procedure.

3.2.5 Intorno complesso e variabile

Quando sono disponibili differenti intorni su cui basare l'algoritmo di ricerca locale, è possibile combinarli in vari modi. In particolare, quando tutti gli intorni vengono esplorati ad ogni passo della ricerca locale, essi formano un intorno complesso; quando invece sono utilizzati alternativamente in modo che se in un intorno non esistono soluzioni miglioranti ne viene esplorato un altro, essi definiscono un intorno variabile.

3.3 Risultati sperimentali

Lo scopo degli esperimenti condotti è stato quello di:

- confrontare la qualità delle soluzioni e i tempi di calcolo in base all'intorno semplice utilizzato per arrivare a definire l'intorno variabile e quello complesso;
- confrontare le prestazioni degli algoritmi di ricerca locale basati sull'intorno variabile e su quello complesso fra di loro e con quelle degli algoritmi basati sugli intorni semplici;
- valutare la robustezza degli algoritmi rispetto all'inizializzazione;

sia nel caso in cui la domanda di un cliente è divisibile sia nel caso in cui non lo è.

Per i test sono state utilizzate le stesse istanze euclidee utilizzate nel capitolo precedente e, a meno che non sia esplicitamente indicato, gli algoritmi di ricerca locale sono inizializzati con le soluzioni calcolate da Farthest Insertion e WEAKSKIP nel caso in cui la domanda sia divisibile e da Farthest Insertion e STRONGSKIP nel caso contrario. Per avere un quadro più completo relativamente alle prestazioni degli algoritmi, in entrambi i casi gli esperimenti sono stati eseguiti due volte: la prima supponendo di avere a disposizione veicoli con capacità 10 per gli insieme 1 e 4, 20 per

gli insiemi 2 e 5, e 40 per insiemi 3 e 6 (veicoli di tipo (a)) e la seconda supponendo di avere a disposizione veicoli con capacità 20 per gli insieme 1 e 4, 40 per gli insiemi 2 e 5, e 80 per insiemi 3 e 6 (veicoli di tipo (b)). I tempi di calcolo riportati sono espressi in secondi.

3.3.1 Domanda indivisibile

Gli esperimenti descritti in questa sottosezione sono stati eseguiti ipotizzando che la domanda dei clienti sia indivisibile.

Intorni semplici

Le tabelle seguenti riportano per ogni insieme di problemi i valori medi dei risultati ottenuti risolvendo le 50 istanze considerando un intorno semplice alla volta; le tabelle 3.1-3.3 sono relative all'utilizzo di veicoli di tipo (a) mentre le tabelle 3.4-3.6 sono relative all'utilizzo di veicoli di tipo (b). I valori migliori sono evidenziati in grassetto.

Essendo gli algoritmi inizializzati con le soluzioni prodotte da STRONGSKIP, che tende a garantire un utilizzo ottimale della capacità dei veicoli all'aumentare di Q (cfr. 2.4.4), la lunghezza media dei cicli che definiscono la soluzione iniziale è circa 4 quando vengono utilizzati veicoli di tipo (a) e circa 8 quando vengono utilizzati i veicoli di tipo (b).

Alla luce dell'osservazione fatta è possibile meglio esaminare i risultati riportati nelle tabelle 3.1 e 3.4. Avendo a disposizione veicoli con capacità limitata (appena pari alla domanda massima di un cliente) la lunghezza dei cicli è tale per cui sono gli algoritmi che utilizzano come intorno CROSS o 3-ARC-EXCHANGE, ed in particolare quest'ultimo, a calcolare le migliori soluzioni. Se solo la capacità dei veicoli è almeno pari al doppio della domanda massima di un cliente, la lunghezza media dei cicli cresce a tal punto che sono gli algoritmi che utilizzano

come intorno EXCHANGEWEAK o RELOCATE, ed in particolare quest'ultimo, a calcolare le soluzioni migliori. L'algoritmo di ricerca locale che utilizza come intorno EXCHANGESTRONG genera sempre soluzioni peggiori di quelle calcolate dall'algoritmo che utilizza EXCHANGEWEAK.

Per quanto riguarda il numero di veicoli utilizzati, dall'esame delle tabelle 3.2 e 3.5 emerge come gli algoritmi che utilizzano EXCHANGEWEAK o EXCHANGESTRONG non possano diminuire il numero di veicoli che caratterizza le soluzioni iniziali; gli algoritmi che si basano su uno degli altri tre intorni riescono invece a considerare tale obiettivo e generano soluzioni che consentono di utilizzare in media fino a due veicoli in meno nel caso in cui essi siano di tipo (a) e un veicolo in meno nel caso in cui essi siano di tipo (b).

Si considerino infine le tabelle 3.3 e 3.6: indipendentemente dal tipo di veicoli utilizzati l'algoritmo che si basa sull'intorno RELOCATE è quello caratterizzato dal minor tempo medio di esecuzione per iterazione, seguito in ordine crescente dagli algoritmi che utilizzano come intorni CROSS, EXCHANGESTRONG, EXCHANGEWEAK e 3-ARC-EXCHANGE. In particolare, l'algoritmo che utilizza quest'ultimo intorno a tempi medi di esecuzione per iterazione di circa un ordine di grandezza più alto rispetto a quelli dell'algoritmo più veloce se lavora sui problemi degli insiemi 1, 2, 3 e di circa due ordini di grandezza più alti se lavora sui problemi degli insiemi 4, 5, 6. I tempi medi di esecuzione complessivi non permettono un ordinamento degli algoritmi in base alla loro velocità perché il numero di iterazioni dipende dall'efficacia degli algoritmi stessi; sono stati tuttavia riportati per dare un'idea del loro ordine di grandezza.

In base all'analisi effettuata e considerando il fatto che l'intorno EXCHANGESTRONG è contenuto in EXCHANGEWEAK, si è deciso di definire

l'intorno complesso utilizzando RELOCATE, EXCHANGEWEAK, CROSS e 3-ARC-EXCHANGE e di definire tre tipi di intorno variabile, v_1, v_{2_a} e v_{2_b}, nel seguente modo:

- in v_1, in base al tempo medio di esecuzione per iterazione del relativo algoritmo, vengono esplorati nell'ordine RELOCATE, CROSS, EXCHANGESTRONG, EXCHANGEWEAK e 3-ARC-EXCHANGE;
- in v_{2_a} e in v_{2_b}, in base alla qualità media delle soluzioni calcolate dal relativo algoritmo utilizzando rispettivamente veicoli di tipo (a) e di tipo (b), vengono esplorati nell'ordine 3-ARC-EXCHANGE, CROSS, EXCHANGEWEAK, RELOCATE e RELOCATE, EXCHANGEWEAK, CROSS, 3-ARC-EXCHANGE.

Intorno complesso

Le tabelle 3.7-3.10 riportano i risultati ottenuti eseguendo la ricerca locale con l'intorno complesso appena definito; le tabelle 3.7 e 3.8 sono relative all'utilizzo di veicoli di tipo (a) mentre la 3.9 e la 3.10 sono relative all'utilizzo di veicoli di tipo (b). In particolare, le ultime otto colonne delle tabelle 3.7 e 3.9 riportano la percentuale media (p) dei passi di ricerca locale effettuati in ciascun intorno e la corrispondente percentuale media (δ) del miglioramento globale ottenuto durante la ricerca. Questi dati permettono di stabilire l'efficacia relativa dell'utilizzo dei vari intorni; i valori migliori sono evidenziati in grassetto.

Rispetto alle migliori soluzioni calcolabili eseguendo un algoritmo di ricerca locale basato su un intorno semplice, la ricerca locale con intorno complesso genera soluzioni caratterizzate da un minor numero medio di veicoli e con un costo medio inferiore di almeno il $3,45\%$ se si utilizzano veicoli di tipo (a) e di almeno il $4,50\%$ se

si utilizzano veicoli di tipo (b); il prezzo da pagare è un elevato tempo medio di esecuzione che si noti però essere dello stesso ordine di grandezza di quello dell'algoritmo che utilizza 3-ARC-EXCHANGE.

Si consideri ora il rapporto $\frac{\delta}{p}$. Per ogni intorno che definisce quello complesso, il suo valore indica l'efficacia relativa dell'utilizzo dell'intorno in questione. Indipendentemente dal tipo di veicoli utilizzati, 3-ARC-EXCHANGE risulta essere l'intorno più efficace, seguito da EXCHANGEWEAK, CROSS e RELOCATE. Questo suggerisce la definizione di un ulteriore intorno variabile, v_3, in cui sono esplorati nell'ordine 3-ARC-EXCHANGE, EXCHANGEWEAK, CROSS e RELOCATE.

Intorni variabili

Le tabelle 3.11-3.14 riportano i risultati ottenuti eseguendo la ricerca locale con ognuno dei quattro intorni variabili definiti. Le tabelle 3.11 e 3.12 sono relative all'utilizzo di veicoli di tipo (a) mentre la 3.13 e la 3.14 sono relative all'utilizzo di veicoli di tipo (b); i valori migliori sono evidenziati in grassetto.

Indipendentemente dal tipo di veicoli è possibile osservare come gli algoritmi che utilizzano come intorno v_1 e v_{2_b} calcolino soluzioni simili sia in termini di costo sia in termini di numero di veicoli utilizzati, e lo fanno impiegando tempi che differiscono poco gli uni dagli altri. Rispetto alla ricerca locale con intorno complesso, questi due algoritmi calcolano soluzioni leggermente peggiori in termini di costo e hanno tempi medi di esecuzione di circa un ordine di grandezza inferiori. È da notare che, comunque, il costo medio delle soluzioni generate dai due algoritmi è almeno il $3,30\%$ e il $3,65\%$ inferiore a quello delle migliori soluzioni calcolabili eseguendo un algoritmo di ricerca locale basato su un intorno semplice, utilizzando rispettivamente veicoli di tipo (a) e di tipo (b).

Anche gli algoritmi che utilizzano come intorno v_{2_a} e v_3 sono caratterizzati da

prestazioni simili; rispetto alla ricerca locale con intorno complesso questi due algoritmi calcolano soluzioni leggermente migliori in termini di costo a fronte di più elevati tempi medi di esecuzione. Questi ultimi, rispetto ai tempi dell'algoritmo di ricerca locale sull'intorno complesso, sono più alti fino ad un massimo del $32,93\%$ se si utilizzano veicoli di tipo (a) e fino ad un massimo del $24,16\%$ se si utilizzano veicoli di tipo (b).

Robustezza rispetto all'inizializzazione

I risultati medi presentati nelle tabelle 3.15-3.18 sono stati ottenuti eseguendo la ricerca locale con gli intorni semplici, quello complesso e quelli variabili a partire dalla soluzione generata da STRONGSKIP quando riceve in ingresso un ciclo hamiltoniano calcolato in maniera casuale. Tra i risultati riportati c'è l'errore relativo (ϵ) espresso in percentuale rispetto ai valori presenti nelle tabelle 3.1, 3.3, 3.7, 3.9, 3.11, 3.13 e la differenza (δ) espressa in percentuale tra il numero medio di veicoli calcolata rispetto ai valori presenti nelle tabelle 3.2, 3.4, 3.8, 3.10, 3.12, 3.14. Le tabelle 3.15 e 3.16 sono relative all'utilizzo di veicoli di tipo (a) mentre la 3.17 e la 3.18 sono relative all'utilizzo di veicoli di tipo (b); i valori migliori sono evidenziati in grassetto.

Le prestazioni degli algoritmi in termini di numero di veicoli utilizzati rimangono pressochè inalterati, indipendentemente dal fatto che i veicoli siano di tipo (a) o (b) e dall'intorno considerato.

Per quanto riguarda il costo delle soluzioni , se i veicoli utilizzati sono di tipo (a), gli unici algoritmi sensibili all'inizializzazione sono quelli che si basano sugli intorni semplici RELOCATE, CROSS e 3-ARC-EXCHANGE, con peggioramenti fino al $5,25\%$. Se invece i veicoli sono di tipo (b), le prestazioni di tutti gli algoritmi degenerano; i più sensibili all'inizializzazione restano gli algoritmi che utilizzano RELOCATE, CROSS e 3-ARC-EXCHANGE, con peggioramenti fino al 15%; quel-

li meno influenzati dall'inizializzazione sono gli algoritmi che si basano sull'intorno complesso e su quelli variabili. Utilizzando v_1 o v_{2_b} le prestazioni peggiorano fino al $3,21\%$ mentre utilizzando l'intorno complesso, v_{2_a} o v_3 il peggioramento è sempre minore del $1,42\%$.

In definitiva, gli algoritmi di ricerca locale basati sull'intorno complesso, v_{2_a} o v_3 sono quelli caratterizzati dai più elevati tempi di esecuzione ma sono anche quelli più robusti rispetto all'inizializzazione e quelli che permettono di ottenere le migliori soluzioni. Rispetto alle soluzioni calcolate da STRONGSKIP, quelle calcolate da questi algoritmi hanno un costo medio inferiore di almeno il $13,68\%$ se si utilizzano veicoli di tipo (a) e di almeno il $16,17\%$ se si utilizzano veicoli di tipo (b); inoltre tali soluzioni prevedono di utilizzare in media fino a due veicoli in meno nel caso in cui essi siano di tipo (a) e un veicolo in meno nel caso in cui essi siano di tipo (b).

3.3.2 Domanda divisibile

Gli esperimenti descritti in questa sottosezione sono stati eseguiti ipotizzando che la domanda dei clienti sia divisibile. In questo caso gli algoritmi di ricerca locale ricevono in ingresso una soluzione calcolata da WEAKSKIP che prevede l'utilizzo del minimo numero di veicoli necessari, la bontà delle soluzioni generate dalla ricerca locale viene quindi discussa solo in termini di costo.

Intorni semplici

Le tabelle seguenti riportano per ogni insieme di problemi i valori medi dei risultati ottenuti risolvendo le 50 istanze considerando un intorno semplice alla volta; le tabelle 3.19 e 3.20 sono relative all'utilizzo di veicoli di tipo(a) mentre la 3.21 e la 3.22 sono relative all'utilizzo di veicoli di tipo (b); i valori migliori sono evidenziati in grassetto.

Dall'esame delle tabelle è possibile notare come, indipendentemente dal tipo di

veicoli utilizzati, l'algoritmo che si basa su RELOCATE sia quello che permette di calcolare le migliori soluzioni e sia quello caratterizzato dal minor tempo di esecuzione medio per iterazione. Rispetto al costo medio delle soluzioni, l'algoritmo è seguito in ordine crescente da quelli che utilizzano come intorni EXCHANGEWEAK, EXCHANGESTRONG, CROSS e 3-ARC-EXCHANGE, mentre rispetto al tempo medio per iterazione è seguito in ordine crescente da quelli che utilizzano come intorni CROSS, EXCHANGESTRONG, EXCHANGEWEAK e 3-ARC-EXCHANGE.

Si considerino ora le due seguenti osservazioni sull'algoritmo che utilizza come intorno 3-ARC-EXCHANGE:

- il suo tempo di esecuzione medio per iterazione, rispetto a quello dell'algoritmo più veloce, è di circa un ordine di grandezza più alto se lavora sui problemi degli insiemi 1, 2, 3 e di circa due ordini di grandezza più alto se lavora sui problemi degli insiemi 4, 5, 6;

- trova soluzioni sempre peggiori di almeno l' $1,60\%$ di quelle calcolate dal miglior algoritmo.

Per la sua dimostrata inefficacia, questo algoritmo non è preso in considerazione nei successivi esperimenti.

In base all'analisi effettuata si è deciso di definire l'intorno complesso utilizzando RELOCATE, EXCHANGEWEAK, CROSS e di definire due tipi di intorno variabile, v_1 e v_2, nel seguente modo:

- in v_1, in base al tempo medio di esecuzione per iterazione del relativo algoritmo, vengono esplorati nell'ordine RELOCATE, CROSS, EXCHANGESTRONG e EXCHANGEWEAK;

- in v_2 , in base alla qualità media delle soluzioni calcolate dal relativo algoritmo vengono esplorati nell'ordine RELOCATE, EXCHANGEWEAK, CROSS.

Intorno complesso

Le tabelle 3.23 e 3.24 riportano i risultati ottenuti eseguendo la ricerca locale con l'intorno complesso appena definito, rispettivamente nei casi in cui siano utilizzati veicoli di tipo (a) e di tipo (b).

Rispetto alle migliori soluzioni calcolabili eseguendo la ricerca locale con un intorno semplice, l'algoritmo che utilizza l'intorno complesso genera soluzioni con un costo medio inferiore di almeno il $2,73\%$ se si utilizzano veicoli di tipo (a) e di almeno il $2,85\%$ se si utilizzano veicoli di tipo (b), a fronte di un aumento del tempo del tempo medio di esecuzione.

Anche sotto l'ipotesi di domanda divisibile, considerando il rapporto $\frac{\delta}{p}$ per ogni intorno, è stato definito un ulteriore intorno variabile, v_3, in cui sono esplorati nell'ordine EXCHANGEWEAK, CROSS e RELOCATE.

Intorni variabili

Le tabelle 3.25 e 3.26 riportano i risultati ottenuti eseguendo la ricerca locale con ognuno dei tre intorni variabili definiti, rispettivamente nei casi in cui siano utilizzati veicoli di tipo (a) o di tipo(b); i valori migliori sono evidenziati in grassetto.

Indipendentemente dal tipo di veicoli utilizzati è possibile osservare come tutti gli algoritmi calcolino soluzioni simili in termini di costo; i valori migliori sono in un terzo dei casi inferiori a quelli che caratterizzano le soluzioni calcolate con l'algoritmo basato sull'intorno complesso.

Per quanto riguarda i tempi medi di esecuzione, quelli degli algoritmi che utiliz-

zano v_1 e v_2 sono circa la metà di quelli degli algoritmi che utilizzano v_3 e l'intorno complesso.

Robustezza rispetto all'inizializzazione

I risultati medi presenti nelle tabelle 3.27 e 3.28 sono stati ottenuti eseguendo la ricerca locale con gli algoritmi semplice, quello complesso e quelli variabili a partire dalla soluzione generata da WEAKSKIP quando riceve in ingresso un ciclo hamiltoniano calcolato in maniera casuale; la tabella 3.27 è relativa all'utilizzo di veicoli di tipo (a) mentre la 3.28 è relativa all'utilizzo di veicoli di tipo (b); i valori migliori sono evidenziati in grassetto.

La ricerca locale con intorni semplici è molto sensibile all'inizializzazione: le soluzioni peggiorano fino ad un massimo del $23,58\%$ se si utilizzano veicoli di tipo (a) e fino ad un massimo del $36,20\%$ se si utilizzano veicoli di tipo (b). Le soluzioni calcolate dagli algoritmi che si basano sull'intorno complesso e su quelli variabili peggiorano invece in maniera più limitata: fino ad un massimo del $8,24\%$ se si utilizzano veicoli di tipo (a) e fino ad un massimo del $9,31\%$ se si utilizzano veicoli di tipo (b).

In definitiva, l'algoritmo di ricerca locale basato sull'intorno complesso è quello caratterizzato dai più elevati tempi di esecuzione ma è anche quello più robusto rispetto all'inizializzazione e quello che permette di ottenere le migliori soluzioni. Rispetto a quelle calcolate da WEAKSKIP, le soluzioni calcolate da questo algoritmo hanno un costo medio inferiore di almeno il $9,5\%$ se si utilizzano veicoli di tipo (a) e di almeno il $13,02\%$ se si utilizzano veicoli di tipo (b).

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	Costo iniziale	Costo finale	Costo finale	Costo finale	Costo finale	Costo finale
1	35,5693	32,2228	32,8072	32,1846	31,9117	31,8013
2	34,7675	31,3159	31,9536	31,3277	30,8846	30,9198
3	34,6180	30,8034	31,7846	30,7767	30,6579	30,4730
4	65,8487	58,8081	59,1244	57,7014	57,6276	57,4708
5	66,3184	58,3176	59,5203	57,8565	57,4957	57,0604
6	66,6556	58,5030	59,6082	57,5099	57,3366	56,7744

Tabella 3.1: domanda indivisibile, veicoli di tipo (a), costo

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	h iniziale	h finale	h finale	h finale	h finale	h finale
1	18,06	17,52	18,06	18,06	17,48	17,42
2	17,44	16,96	17,44	17,44	16,96	16,90
3	16,80	16,36	16,80	16,80	16,32	16,30
4	33,48	32,34	33,48	33,48	32,38	32,20
5	33,04	31,74	33,04	33,04	31,76	31,60
6	32,66	31,84	32,66	32,66	31,88	31,64

Tabella 3.2: domanda indivisibile, veicoli di tipo (a), numero di veicoli

	RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X	
Set	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.
1	0,012	0,0006	0,016	0,0009	0,040	0,0021	0,015	0,0007	0,292	0,0205
2	0,013	0,0007	0,016	0,0009	0,042	0,0022	0,014	0,0007	0,268	0,0197
3	0,013	0,0007	0,017	0,0010	0,049	0,0025	0,014	0,0007	0,300	0,0191
4	0,055	0,0014	0,087	0,0020	0,211	0,0046	0,135	0,0028	5,265	0,1678
5	0,065	0,0015	0,091	0,0021	0,246	0,0051	0,138	0,0028	5,451	0,1647
6	0,065	0,0015	0,091	0,0021	0,260	0,0054	0,141	0,0028	5,842	0,1648

Tabella 3.3: domanda indivisibile, veicoli di tipo (a), tempi di esecuzione

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	Costo iniziale	Costo finale	Costo finale	Costo finale	Costo finale	Costo finale
1	20,6681	18,2135	19,5086	18,5099	19,2083	19,2400
2	19,9296	17,5037	18,9002	17,7919	18,6323	18,6956
3	20,0962	17,4820	18,8468	17,7800	18,5534	18,6373
4	36,7439	32,3693	34,2802	32,3223	33,8364	33,8869
5	37,0301	31,7270	34,3082	32,0785	33,6385	33,6058
6	37,0050	31,5725	34,2782	31,8121	33,5005	33,5295

Tabella 3.4: domanda indivisibile, veicoli di tipo (b), costo

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	h iniziale	h finale	h finale	h finale	h finale	h finale
1	8,92	8,70	8,92	8,92	8,66	8,68
2	8,50	8,26	8,50	8,50	8,24	8,20
3	8,22	8,02	8,22	8,22	8,00	8,02
4	16,24	15,70	16,24	16,24	15,76	15,72
5	15,76	15,32	15,76	15,76	15,26	15,22
6	15,46	15,24	15,46	15,46	15,14	15,16

Tabella 3.5: domanda indivisibile, veicoli di tipo (b), numero di veicoli

	RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X	
Set	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.
1	0,017	0,0006	0,023	0,0019	0,100	0,0062	0,005	0,0006	0,074	0,0104
2	0,014	0,0007	0,023	0,0020	0,103	0,0068	0,004	0,0006	0,065	0,0099
3	0,016	0,0007	0,027	0,0021	0,118	0,0076	0,005	0,0006	0,072	0,0096
4	0,064	0,0016	0,115	0,0042	0,524	0,0142	0,042	0,0020	1,410	0,0971
5	0,076	0,0016	0,121	0,0044	0,584	0,0154	0,045	0,0020	1,559	0,0953
6	0,075	0,0016	0,125	0,0046	0,691	0,0170	0,045	0,0020	1,456	0,0947

Tabella 3.6: domanda indivisibile, veicoli di tipo (b), tempi di esecuzione

				RELOCATE		XWEAK		CROSS		3-ARC-X	
Set	Costo iniziale	Costo finale	Ex. time	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)
1	35,5693	30,7029	0,594	34	23	31	25	8	4	28	48
2	34,7675	29,5737	0,582	37	24	28	22	9	7	26	48
3	34,6180	29,1772	0,596	34	20	30	28	7	4	29	48
4	65,8487	55,1963	9,415	29	15	32	25	8	4	31	56
5	66,3184	54,6056	10,239	31	18	30	22	7	3	32	57
6	66,6556	54,4594	9,964	32	18	30	23	7	4	31	55

Tabella 3.7: domanda indivisibile, veicoli di tipo (a), intorno complesso, costo e tempo di esecuzione

Set	h iniziale	h finale
1	18,06	17,34
2	17,44	16,80
3	16,80	16,26
4	33,48	32,18
5	33,04	31,48
6	32,66	31,62

Tabella 3.8: domanda indivisibile, veicoli di tipo (a), intorno complesso, numero di veicoli

				RELOCATE		XWEAK		CROSS		3-ARC-X	
Set	Costo iniziale	Costo finale	Ex. time	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)
1	20,6681	17,3241	0,412	39	25	45	47	8	8	9	20
2	19,9296	16,6862	0,396	41	28	45	47	7	7	7	17
3	20,0962	16,5034	0,427	39	26	46	50	6	7	9	17
4	36,7439	30,4447	5,512	35	22	47	47	7	6	11	26
5	37,0301	29,9889	5,842	36	22	47	47	6	6	11	25
6	37,0050	29,7755	5,806	37	23	47	46	5	5	10	26

Tabella 3.9: domanda indivisibile, veicoli di tipo (b) , intorno complesso, costo e tempo di esecuzione

Set	h iniziale	h finale
1	8,92	8,66
2	8,50	8,22
3	8,22	7,96
4	16,24	15,70
5	15,76	15,20
6	15,46	15,12

Tabella 3.10: domanda indivisibile, veicoli di tipo (b), intorno complesso, numero di veicoli

		v_1		v_{2a}		V_{2b}		v_3	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	35,5693	30,7272	0,068	30,6810	0,666	30,7097	0,083	30,7193	0,627
2	34,7675	29,6384	0,065	29,4905	0,668	29,7049	0,073	29,4704	0,633
3	34,6180	29,2252	0,064	29,1070	0,659	29,1850	0,081	29,1309	0,635
4	65,8487	55,4154	0,503	55,0602	12,455	55,4038	0,565	55,0478	11,976
5	66,3184	55,0224	0,503	54,5341	12,671	54,9729	0,567	54,5175	13,125
6	66,6556	54,9005	0,568	54,3916	13,245	54,7666	0,633	54,3655	12,348

Tabella 3.11: domanda indivisibile, veicoli di tipo (a), intorni variabili, costo e tempo di esecuzione

		v_1	v_{2a}	v_{2b}	v_3
Set	h iniziale	h finale	h finale	h finale	h finale
1	18.06	17,34	17,32	17,38	17,32
2	17.44	16,78	16,80	16,82	16,76
3	16.8	16,24	16,22	16,26	16,22
4	33.48	32,24	32,16	32,30	32,14
5	33.04	31,58	31,40	31,58	31,40
6	32.66	31,68	31,60	31,64	31,60

Tabella 3.12: domanda indivisibile, veicoli di tipo (a), intorni variabili, numero di veicoli

		V_1		v_{2a}		v_{2b}		v_3	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	20,6681	17,5478	0,059	17,2608	0,401	17,5314	0,071	17,3101	0,386
2	19,9296	16,7987	0,063	16,6656	0,389	16,8181	0,071	16,6472	0,406
3	20,0962	16,6603	0,066	16,5321	0,421	16,6383	0,082	16,4921	0,424
4	36,7439	30,7216	0,372	30,3254	6,771	30,8449	0,411	30,3775	6,605
5	37,0301	30,2615	0,354	29,8935	7,048	30,1906	0,472	29,8127	7,412
6	37,0050	29,9904	0,371	29,7418	7,091	30,0228	0,447	29,6533	7,213

Tabella 3.13: domanda indivisibile, veicoli di tipo (b), intorni variabili, costo e tempo di esecuzione

		v_1	v_{2a}	v_{2b}	v_3
Set	h iniziale	h finale	h finale	h finale	h finale
1	8,92	8,68	8,56	8,66	8,62
2	8,50	8,24	8,20	8,22	8,18
3	8,22	7,98	7,98	7,94	7,96
4	16,24	15,64	15,66	15,66	15,66
5	15,76	15,20	15,12	15,14	15,10
6	15,46	15,14	15,08	15,14	15,06

Tabella 3.14: domanda indivisibile, veicoli di tipo (b), intorni variabili, numero di veicoli

		RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X		Complesso		V_1		V_{2a}		V_{2b}		V_3	
Set	Costo iniziale	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)
1	41,4564	33,0284	2,50	33,0482	0,73	32,1754	-0,03	32,4117	1,57	32,3231	1,64	30,6884	-0,05	30,8176	0,29	30,5848	-0,31	30,8258	0,38	30,6549	-0,10
2	41,0221	32,3661	3,35	32,2058	0,79	31,3247	-0,01	31,8213	3,03	31,9301	3,27	29,6805	0,36	29,8863	0,84	29,6808	0,65	29,9396	0,79	29,6130	0,43
3	40,4861	31,6246	2,67	31,7589	-0,08	30,9595	0,59	31,1882	1,73	31,1626	2,26	29,1378	-0,14	29,4003	0,60	29,1199	0,04	29,2863	0,35	29,1647	0,11
4	81,4759	61,8977	5,25	59,2565	0,22	57,7792	0,13	58,9437	2,28	58,5258	1,84	55,1620	-0,06	55,4202	0,01	55,1152	0,10	55,3472	-0,10	55,0410	0,05
5	81,1059	60,6285	3,96	59,2130	-0,52	57,4909	-0,63	58,2038	1,23	57,8491	1,38	54,6246	0,03	54,9606	-0,11	54,6050	0,13	54,8807	-0,17	54,5330	0,02
6	80,5319	60,5302	3,47	59,2632	-0,58	57,5166	0,01	57,9742	1,11	57,4919	1,26	54,6609	0,37	55,0544	0,28	54,3612	-0,06	54,8091	0,08	54,3951	0,07

Tabella 3.15: robustezza rispetto all'inizializzazione, domanda indivisibile, veicoli di tipo (a), costo

		RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X		Complesso		V_1		V_{2a}		V_{2b}		V_3	
Set	h iniziale	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)
1	17,94	17,44	-0,46	17,94	-0,66	17,94	-0,66	17,34	-0,80	17,36	-0,34	17,38	0,23	17,30	-0,23	17,26	-0,35	17,30	-0,46	17,26	-0,23
2	17,36	16,82	-0,83	17,36	-0,46	17,36	-0,46	16,84	-0,71	16,82	-0,47	16,80	0,00	16,78	0,00	16,70	-0,60	16,74	-0,48	16,70	-0,36
3	16,98	16,28	-0,49	16,98	1,07	16,98	1,07	16,28	-0,25	16,18	-0,74	16,14	-0,74	16,12	-0,74	16,06	-0,99	16,12	-0,86	16,12	-0,62
4	33,26	32,22	-0,37	33,26	-0,66	33,26	-0,66	32,42	0,12	32,22	0,06	32,22	0,12	32,10	-0,43	32,12	-0,12	32,14	-0,50	32,14	0,00
5	32,28	31,40	-1,07	32,28	-2,30	32,28	-2,30	31,56	-0,63	31,40	-0,63	31,30	-0,57	31,30	-0,89	31,28	-0,38	31,26	-1,01	31,30	-0,38
6	32,48	31,58	-0,82	32,48	-0,55	32,48	-0,55	31,66	-0,69	31,52	-0,38	31,48	-0,44	31,44	-0,76	31,32	-0,89	31,40	-0,76	31,36	-0,70

Tabella 3.16: robustezza rispetto all'inizializzazione, domanda indivisibile, veicoli di tipo (a), numero di veicoli

		RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X		Complesso		V_1		V_{2a}		V_{2b}		V_3	
Set	Costo iniziale	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)	Costo finale	Δ(%)
1	31,8289	20,0090	9,86	20,5071	5,12	18,9515	2,39	21,7407	13,18	21,8853	13,75	17,5696	1,42	17,7823	1,34	17,4225	0,94	17,9273	2,26	17,3749	0,59
2	31,4910	19,2794	10,14	19,9943	5,79	18,3981	3,41	21,1429	13,47	21,4399	14,68	16,9145	1,37	17,2189	2,50	16,8026	0,82	17,2139	2,35	16,8768	1,36
3	31,1353	18,9177	8,21	19,8509	5,33	18,0569	1,56	20,6655	11,38	20,8375	11,81	16,6172	0,69	17,1104	2,70	16,5402	0,05	16,9313	1,76	16,5268	0,28
4	63,4016	36,6498	13,22	36,3954	6,17	33,2830	2,97	38,3639	13,38	37,7065	11,27	30,7650	1,05	31,4591	2,40	30,5297	0,67	31,5078	2,15	30,6323	0,93
5	63,4856	35,8133	12,88	36,6670	6,88	33,0157	2,92	37,1971	10,58	37,1713	10,61	30,3315	1,14	31,2023	3,11	30,0334	0,47	31,1606	3,21	30,2025	1,27
6	62,8485	35,2652	11,70	35,9323	4,83	32,4084	1,87	36,8169	9,90	36,2091	7,99	30,0730	1,00	30,7147	2,42	29,7518	0,03	30,9715	3,16	29,8495	0,81

Tabella 3.17: robustezza rispetto all'inizializzazione, domanda indivisibile, veicoli di tipo (b), costo

		RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X		Complesso		V_1		V_{2a}		V_{2b}		V_3	
Set	h iniziale	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)	h finale	Δ(%)
1	8,80	8,58	-1,38	8,80	-1,35	8,80	-1,35	8,58	-0,92	8,60	-0,92	8,50	-1,85	8,52	-1,84	8,56	0,00	8,56	-1,15	8,54	-0,70
2	8,48	8,20	-0,73	8,48	-0,24	8,48	-0,24	8,20	-0,49	8,24	0,49	8,16	-0,73	8,12	-1,46	8,16	-0,49	8,14	-0,97	8,14	-0,73
3	8,24	8,00	-0,25	8,24	0,24	8,24	0,24	8,02	0,25	8,06	0,50	7,98	0,25	7,98	0,00	7,98	0,00	7,98	0,50	8,00	0,50
4	16,16	15,80	0,64	16,16	-0,49	16,16	-0,49	15,80	0,25	15,90	1,15	15,70	0,00	15,70	0,38	15,76	0,64	15,62	-0,26	15,74	0,51
5	15,64	15,38	0,39	15,64	-0,76	15,64	-0,76	15,34	0,52	15,34	0,79	15,20	0,00	15,26	0,39	15,24	0,79	15,26	0,79	15,20	0,66
6	15,64	15,28	0,26	15,64	1,16	15,64	1,16	15,20	0,40	15,22	0,40	15,14	0,13	15,16	0,13	15,10	0,13	15,10	-0,26	15,14	0,53

Tabella 3.18: robustezza rispetto all'inizializzazione, domanda indivisibile, veicoli di tipo (b), numero di veicoli

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	Costo iniziale	Costo finale	Costo finale	Costo finale	Costo finale	Costo finale
1	33,4204	30,7701	31,3098	30,8147	31,4166	31,5494
2	32,4415	29,8364	30,5313	30,0348	30,7677	30,9227
3	31,9873	29,3302	30,0529	29,4970	30,5803	30,5236
4	59,6457	55,7025	56,0377	55,1315	56,3666	56,6042
5	58,9753	54,8829	55,6806	54,7269	55,9942	56,0490
6	58,0407	54,0460	55,0372	54,1479	55,4663	55,5463

Tabella 3.19: domanda divisibile, veicoli di tipo (a), costo

	RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X	
Set	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.
1	0,022	0,0009	0,037	0,0019	0,090	0,0047	0,019	0,0011	0,430	0,0405
2	0,022	0,0009	0,039	0,0021	0,089	0,0052	0,016	0,0011	0,387	0,0415
3	0,023	0,0009	0,040	0,0023	0,095	0,0057	0,014	0,0011	0,402	0,0409
4	0,089	0,0019	0,184	0,0043	0,457	0,0108	0,153	0,0045	7,258	0,3519
5	0,094	0,0020	0,183	0,0048	0,451	0,0121	0,134	0,0046	7,261	0,3578
6	0,099	0,0020	0,175	0,0049	0,423	0,0123	0,127	0,0047	6,676	0,3659

Tabella 3.20: domanda divisibile, veicoli di tipo (a), tempi di esecuzione

		RELOCATE	XSTRONG	XWEAK	CROSS	3-ARC-X
Set	Costo iniziale	Costo finale	Costo finale	Costo finale	Costo finale	Costo finale
1	20,2372	17,9184	18,8911	17,9784	19,3253	19,4236
2	19,6512	17,4283	18,4279	17,5787	18,9274	18,9927
3	19,3812	17,0894	18,3223	17,3568	18,7175	18,7302
4	34,8967	31,3953	32,6986	31,2523	33,5155	33,5801
5	34,5411	30,9864	32,5422	31,1168	33,1685	33,1996
6	34,0453	30,4917	31,9875	30,6806	32,8363	32,8965

Tabella 3.21: domanda divisibile, veicoli di tipo (b), costo

	RELOCATE		XSTRONG		XWEAK		CROSS		3-ARC-X	
Set	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.	Ex. time	Ex. time per iter.
1	0,019	0,0008	0,041	0,0029	0,161	0,0101	0,005	0,0007	0,092	0,0158
2	0,019	0,0008	0,041	0,0031	0,176	0,0113	0,004	0,0006	0,072	0,0153
3	0,020	0,0009	0,041	0,0033	0,204	0,0131	0,004	0,0007	0,075	0,0162
4	0,081	0,0017	0,188	0,0063	0,776	0,0225	0,034	0,0027	1,397	0,1526
5	0,084	0,0019	0,194	0,0069	0,764	0,0254	0,034	0,0027	1,437	0,1545
6	0,089	0,0019	0,420	0,0133	0,804	0,0262	0,032	0,0027	1,341	0,1547

Tabella 3.22: domanda divisibile, veicoli di tipo (b), tempi di esecuzione

				RELOCATE		XWEAK		CROSS	
Set	Costo iniziale	Costo finale	Ex. time	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)
1	33,4204	29,7311	0,211	36	26	47	58	17	16
2	32,4415	29,0222	0,207	39	25	46	61	15	14
3	31,9873	28,4836	0,218	41	28	46	60	14	12
4	59,6457	53,6212	1,087	29	19	52	62	19	20
5	58,9753	52,9135	1,110	34	23	48	56	18	21
6	58,0407	52,5240	1,084	38	25	45	56	18	19

Tabella 3.23: domanda divisibile, ve icoli di tipo (a) , intorno compl esso, costo e tempo di esecuzione

				RELOCATE		XWEAK		CROSS	
Set	Costo iniziale	Costo finale	Ex. time	p(%)	/(%)	p(%)	/(%)	p(%)	/(%)
1	20,2372	17,3487	0,264	36	21	55	70	9	9
2	19,6512	16,7873	0,283	39	27	53	66	7	8
3	19,3812	16,6025	0,328	35	23	58	70	7	7
4	34,8967	30,3387	1,250	29	16	63	73	9	11
5	34,5411	29,9808	1,312	33	19	59	69	8	12
6	34,0453	29,6146	1,407	35	21	57	69	8	10

Tabella 3.24: domanda divisibile, veicoli di tipo (b) , intorno complesso, costo e tempo di esecuzione

		v_1		v_2		v_3	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	33,4204	29,8443	0,078	29,8453	0,094	29,7667	0,175
2	32,4415	29,0243	0,067	29,0040	0,085	29,0808	0,173
3	31,9873	28,5101	0,069	28,5026	0,085	28,5358	0,200
4	59,6457	53,7240	0,396	53,7880	0,488	53,7322	0,910
5	58,9753	53,0698	0,357	53,1407	0,460	53,0635	0,940
6	58,0407	52,6326	0,320	52,6954	0,407	52,6057	0,901

Tabella 3.25: domanda divisibile, ve icoli di tipo (a), intorni vari abili, costo e tempo di esecuzione

		v_1		v_2		v_3	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	20,2372	17,3979	0,073	17,4246	0,083	17,4259	0,269
2	19,6512	16,8323	0,078	16,8823	0,085	16,8740	0,312
3	19,3812	16,5860	0,082	16,5525	0,097	16,6473	0,376
4	34,8967	30,3216	0,349	30,4101	0,423	30,4381	1,297
5	34,5411	30,0944	0,353	30,1300	0,426	30,1049	1,412
6	34,0453	29,7356	0,328	29,7907	0,374	29,5958	1,526

Tabella 3.26: domanda divisibile, ve icoli di tipo (b), intorni vari abili, costo e tempo di esecuzione

		RELOCATE		XSTRONG		XWEAK		CROSS		Complesso		V_1		V_2		V_3	
Set	Costo iniziale	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)
1	49,5891	35,5255	15,45	34,3440	9,69	32,8349	6,56	34,5089	9,84	30,7023	3,27	31,0120	3,91	30,7971	3,19	30,6091	2,83
2	49,3065	34,9650	17,19	35,2659	15,51	33,2524	10,71	35,7608	16,23	30,2968	4,39	30,6654	5,65	30,5806	5,44	30,4429	4,68
3	49,7165	34,4496	17,45	35,4970	18,12	33,7815	14,53	36,1102	18,08	30,5821	7,37	30,6327	7,45	30,6440	7,51	30,6775	7,51
4	98,3485	67,5996	21,36	62,1353	10,88	59,3116	7,58	63,8269	13,24	55,2052	2,95	55,9526	4,15	55,8096	3,76	55,4479	3,19
5	99,1983	67,0539	22,18	64,9196	16,59	60,8937	11,27	65,8800	17,66	55,8198	5,49	56,4127	6,30	56,5780	6,47	56,0722	5,67
6	100,1700	66,7910	23,58	66,8180	21,41	62,2118	14,89	68,0260	22,64	56,4908	7,55	56,9676	8,24	56,8362	7,86	56,8640	8,09

Tabella 3.27: robustezza rispetto all'inizializzazione, domanda divisibile, veicoli di tipo (a), costo

		RELOCATE		XSTRONG		XWEAK		CROSS		Complesso		V_1		V_2		V_3	
Set	Costo iniziale	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)	Costo finale	(%)
1	36,3819	20,8625	16,43	22,0157	16,54	19,4746	8,32	23,3758	20,96	18,0023	3,77	18,2127	4,68	18,1523	4,18	18,1105	3,93
2	36,1289	20,3846	16,96	22,1992	20,47	19,4200	10,47	24,2593	28,17	17,6077	4,89	17,9133	6,42	17,9691	6,44	17,7588	5,24
3	36,5005	20,3193	18,90	22,1270	20,77	19,3448	11,45	24,4436	30,59	17,6724	6,44	17,7243	6,86	17,8760	8,00	17,8024	6,94
4	73,0382	38,9002	23,90	39,4982	20,79	34,5502	10,55	42,7358	27,51	31,9136	5,19	32,2418	6,33	32,5083	6,90	31,9876	5,09
5	73,4665	38,6192	24,63	40,6739	24,99	35,1965	13,11	43,7113	31,79	31,7205	5,80	32,5750	8,24	32,4308	7,64	32,1408	6,76
6	74,0461	38,3543	25,79	41,0965	28,48	35,1738	14,65	44,7220	36,20	31,852	7,56	32,3790	8,89	32,5645	9,31	32,0960	8,45

Tabella 3.28: robustezza rispetto all'inizializzazione, domanda divisibile, veicoli di tipo (b), costo

Capitolo 4

Tabu search

4.1 Descrizione generale dell'algoritmo

Il tabu search (TS) è un algoritmo di ricerca locale in cui si permette alla funzione obiettivo di peggiorare; in questo modo la ricerca non termina appena viene trovato un minimo locale e lo spazio delle soluzioni del problema in questione viene ampiamente esplorato. Per evitare che l'algoritmo continui a considerare sempre le stesse soluzioni, quelle recentemente esaminate vengono inserite in una lista delle soluzioni proibite (tabu list) e vi rimangono per un certo numero di iterazioni; un passo di ricerca locale che porta a considerare una soluzione presente nella tabu list è detto mossa tabu. Tipicamente la ricerca termina dopo che è stato eseguito un determinato numero di iterazioni senza ottenere dei miglioramenti o più semplicemente dopo un numero fisso di iterazioni. Il tabu search è una tecnica generica i cui parametri di funzionamento devono essere opportunamente tarati a seconda del tipo di problema. Per una descrizione più esauriente dell'algoritmo si faccia riferimento a [19].

Nella letteratura scientifica relativa ai VRPs molti ricercatori hanno riportato buoni risultati ottenuti attraverso il TS; per es. l'algoritmo è stato applicato con successo alla risoluzione del TSPSPD in [20] e del CVRP in [21] e [22].

4.2 Algoritmi TS per il VRPSPD

Si tenga innanzitutto presente la seguente osservazione relativa all'implementazione di un TS. Per rendere efficiente l'algoritmo, invece di memorizzare l'intera soluzione è possibile memorizzare alcuni suoi attributi e considerare tabu tutte quelle soluzioni in possesso di tali attributi.

Basandosi su questa osservazione e sugli intorni definiti nel capitolo precedente (non considerando EXCHANGESTRONG perché contenuto in EXCHANGEWEAK), sono state realizzate varie implementazioni del TS per il VRPSPD che hanno in comune le seguenti caratteristiche:

- ad ogni iterazione effettuano un passo di ricerca locale che porta sempre a considerare soluzioni ammissibili;
- effettuano passi di ricerca locale verso soluzioni tabu (soluzioni ammissibili i cui attributi appartengono alla tabu list) solo se il loro costo e minore di quello della migliore soluzione trovata fino a quel momento;
- terminano dopo T iterazioni oppure quando a partire dalla soluzione corrente non è più possibile trovare soluzioni ammissibili non tabu .

Nel seguito viene descritta, per ogni algoritmo, la scelta degli attributi delle soluzioni da memorizzare (in tali descrizioni vengono utilizzate le notazioni introdotte in 3.1).

4.2.1 TS_{REL} e TS_{XWEAK}

TS_{REL} e TS_{XWEAK} si basano rispettivamente sugli intorni RELOCATE e EXCHANGEWEAK definiti in 3.2.1 e 3.2.2. In TS_{REL}, ad ogni iterazione, se il

passo di ricerca prevede che la visita $v_{t_1,k_1} = (i, r_i)$ venga spostata dal ciclo t_1 e reinserita in un altro ciclo, è memorizzata nella tabu list la coppia di attributi (t_1, i): ogni mossa che nelle successive iterazioni reinserisce il cliente i nel ciclo t_1 è tabu. In TS_{XWEAK} invece, ad ogni iterazione in cui la visita $v_{t_1,k_1} = (i, r_i)$ viene scambiata con la visita $v_{t_2,k_2} = (j, r_j)$, sono memorizzate nella tabu list le coppie di attribuiti (t_1, i) e (t_2, j): ogni mossa che nelle successive iterazioni reinserisce o il cliente i in nel ciclo t_1 o il cliente j nel ciclo t_2 è tabu.

4.2.2 TS_{CROSS} e TS_{3ARCX}

TS_{CROSS} e TS_{3ARCX} utilizzano rispettivamente gli intorni CROSS e 3-ARC-EXCHANGE definiti in 3.2.3 e 3.2.4. In TS_{CROSS}, ad ogni iterazione in cui gli archi (i_1, j_1) e (i_2, j_2) vengono sostituiti dagli archi (i_1, j_2) e (i_2, j_1), sono memorizzate nella tabu list le coppie di attributi (i_1, j_1) e (i_2, j_2): ogni mossa che nelle successive iterazioni reinserisce almeno uno degli archi eliminati è proibita. Operazioni analoghe avvengono in TS_{3ARCX}; in particolare, essendo tre gli archi sostituiti ad ogni passo di ricerca locale, sono tre le coppie di attributi inserite nella tabu list ad ogni iterazione.

4.2.3 TS_c e TS_v

Osman [23] presenta due implementazioni del TS per il CVRP basate sull'utilizzo di tre intorni. Nell'implementazione dell'algoritmo chiamata BA (Best Admissible) ad ogni passo di ricerca locale vengono esplorati tutti e tre gli intorni e viene scelta la soluzione ammissibile migliore non tabu. Nell'altra implementazione, FBA (First Best Admissible), viene scelta la prima soluzione ammissibile migliorante non tabu trovata esplorando in un ordine prefissato i tre intorni, se tutte le soluzioni ammissibili non tabu sono peggioranti viene selezionata la migliore. I risultati riportati in [23] mostrano come queste due implementazioni producano eccellenti risultati.

TS_c e TS_v sono implementazioni del tabu search per il VRPSPD, simili rispettivamente a BA e FBA, basate sull'utilizzo di RELOCATE, EXCHANGEWEAK, CROSS e 3-ARC-EXCHANGE.

Per meglio comprendere il funzionamento di TS_c occorre distinguere le due fasi che caratterizzano il processo di ricerca adottato nelle implementazioni del tabu search fin qui descritte:

- la fase (1) in cui ad ogni iterazione la soluzione migliora fino al raggiungimento di un minimo locale;
- la fase (2) in cui la soluzione peggiora sempre più ad ogni passo di ricerca locale fino al raggiungimento dell'intorno di un minimo locale (auspicabilmente diverso da quelli precedentemente raggiunti).

La fase (1) termina quando, attraverso un passo di ricerca, non è possibile raggiungere una soluzione migliorante non tabu; la fase (2) termina quando viene trovata una soluzione migliorante non tabu o una soluzione tabu che ha un costo minore di quello della soluzione migliore trovata fino a quel momento. In TS_c, nella fase (1) vengono esplorati tutti gli intorni e viene scelta la migliore soluzione ammissibile non tabu. Nella fase (2) , per scegliere la migliore soluzione ammissibile non tabu, viene invece esplorato solo un intorno U fra quelli a disposizione. All'inizio della fase (2), l'intorno U è quello in cui è stata trovata la soluzione ammissibile peggiorante non tabu che ha stabilito la terminazione della fase (1); se in tale intorno non sono più presenti soluzioni ammissibili non tabu, l'intorno U diventa di volta in volta quello che fra i rimanenti contiene la migliore soluzione ammissibile non tabu.

Il funzionamento di TS_v è invece più intuitivo: ad ogni iterazione viene scelta la prima soluzione ammissibile migliorante non tabu trovata esplorando gli intorni nell'ordine specificato nella sezione 4.4; se tutte le soluzioni ammissibili non tabu

sono peggioranti, viene selezionata la prima il cui costo differisce da quello della soluzione corrente per un valore inferiore a σ; infine, se tutte il costo di tutte soluzione ammissibili peggioranti non tabu differisce da quello della soluzione corrente per un valore superiore a σ, viene selezionata la migliore.

Verificare se una mossa è tabu o meno è un'operazione che viene eseguita in maniera diversa a seconda dell'intorno in cui si sta cercando di effettuare il passo di ricerca locale:

- se l'intorno è RELOCATE, spostare la visita $v_{t_1,k_1} = (i_1, r_{i_1})$ dal ciclo t_1, reinserendola in un ciclo t_2 fra le visite $v_{t_2,k_2} = (i_2, r_{i_2})$ e $v_{t_2,k_2+1} = (j_2, r_{j_2})$, è una mossa tabu se è presente nella tabu list almeno una fra le coppie di attributi (t_2, i_1), (i_2, i_1), (i_1, j_2);

- se l'intorno è EXCHANGEWEAK, scambiare fra di loro le visite $v_{t_1,k_1} = (i_1, r_{i_1})$ e $v_{t_2,k_2} = (i_2, r_{i_2})$ reinserendole rispettivamente fra le visite $v_{t_2,k_2-1} = (p_2, r_{p_2})$, $v_{t_2,k_2+1} = (s_2, r_{s_2})$ e $v_{t_1,k_1-1} = (p_1, r_{p_1})$, $v_{t_1,k_1+1} = (s_1, r_{s_1})$ è una mossa tabu se è presente nella tabu list almeno una delle coppie di attributi (t_2, i_1), (t_1, i_2), (p_2, i_1), (i_1, s_2), (p_1, i_2), (i_2, s_1);

- se l'intorno è CROSS, sostituire gli archi (i_1, j_1), (i_2, j_2), facenti parte dei cicli t_1 e t_2, con gli archi (i_1, j_2), (i_2, j_1) è una mossa tabu se è presente nella tabu list almeno una fra le coppie di attributi (t_1, j_2), (t_2, j_1), (i_1, j_2), (i_2, j_1);

- se l'intorno è 3-ARC-EXCHANGE, sostituire gli archi (i_1, j_1), (i_2, j_2), (i_3, j_3), facenti parte dei cicli t_1, t_2 e t_3, con gli archi (i_1, j_2), (i_2, j_3), (i_3, j_1) è una mossa tabu se è presente nella tabu list almeno una fra le coppie di attributi (t_1, j_2), (t_2, j_3), (t_3, j_1), (i_1, j_2), (i_2, j_3), (i_3, j_1); sostituire gli archi (i_1, j_1), (i_2, j_2), (i_3, j_3) con gli archi (i_1, j_3), (i_2, j_1), (i_3, j_2) è una mossa tabu se è pre-

sente nella tabu list almeno una fra le coppi di attributi (t_1, j_3), (t_2, j_1), (t_3, j_2), (i_1, j_3), (i_2, j_1), (i_3, j_2).

4.3 Dettagli implementativi

Il numero massimo di iterazioni effettuate dagli algoritmi è pari a 2000; nel seguito vengono descritte, per ognuno di essi, l'implementazione, l'inizializzazione e il controllo della lunghezza della tabu list.

Negli algoritmi TS che utilizzano un unico intorno (TS_{REL}, TS_{XWEAK}, TS_{CROSS} e TS_{3ARCX}) la tabu list è implementata attraverso una matrice che memorizza per ogni coppia la più recente iterazione in cui è stata inclusa. In TS_c e TS_v, invece, la tabu list è implementata attraverso due matrici, m_1 e m_2, in cui vengono rispettivamente memorizzate le più recenti iterazioni in cui le coppie di attributi (t, i) sono state incluse a causa di passi di ricerca locale effettuati o in RELOCATE o in EXCHANGEWEAK, e le più recenti iterazioni in cui le coppie di attributi (i, j) sono state incluse a causa di passi di ricerca effettuati o in CROSS o in 3-ARC-EXCHANGE. Ogni elemento delle matrici è inizializzato a $-\infty$. L'utilizzo delle matrici consente di verificare se una mossa è tabu o meno in tempo costante.

La lunghezza della tabu list è determinata grazie ad un parametro l: all'iterazione n-esima sono presenti nella tabu list tutte le coppie per cui $n - n_c \leq l$, dove n_c indica l'iterazione più recente in cui la coppia c è stata inclusa. Il valore di l viene fatto variare in un 'intervallo $[l_{min}, l_{max}]$ nel seguente modo:

- se viene fatta una mossa migliorante (un passo di ricerca locale che porta a considerare una soluzione migliore di quella attuale) il nuovo valore di l è $max\,\{l - 1, l_{min}\}$;
- se viene fatta una mossa peggiorante il nuovo valore di l è $min\,\{l + 1, l_{max}\}$.

In particolare, in TS_c e TS_v, il numero di coppie di attributi memorizzate nella matrice m_1 va da un minimo di $l_{1_{min}}$ ad un massimo di $l_{1_{max}}$ ed il numero di quelle memorizzate nella matrice m_2 va da un minimo di $l_{2_{min}}$ ad un massimo di $l_{2_{max}}$, perciò $l_{min} = l_{1_{min}} + l_{2_{min}}$ e $l_{max} = l_{1_{max}} + l_{2_{max}}$.

Per ogni TS continuano a valere le analisi delle operazioni di generazione, controllo dell'ammissibilità, valutazione e aggiornamento relative all'intorno utilizzato.

4.4 Risultati sperimentali

Lo scopo degli esperimenti svolti è stato quello di:

- confrontare fra di loro le prestazioni dei vari algoritmi TS;
- valutare la qualità delle soluzioni calcolate dalla migliore implementazione del TS;

nel caso in cui la domanda di un cliente è indivisibile.

Considerando i tempi medi per iterazione degli algoritmi di ricerca locale basati su intorni semplici (cfr. 3.3.1), si è deciso che in TS_v vengano esplorati nell'ordine RELOCATE, CROSS, EXCHANGEWEAK e 3-ARC-EXCHANGE. Inoltre, considerando l'elevato tempo medio di esecuzione per iterazione dell'algoritmo che effettua la ricerca locale basandosi su 3-ARC-EXCHANGE, sono state valutate due versioni di TS_c: TS_{c_1} e TS_{c_2}; nella prima versione 3-ARC-EXCHANGE non viene esplorato mentre nella seconda sì.

Per i test sono state utilizzate le stesse istanze euclidee utilizzate nei capitoli precedenti; gli algoritmi TS sono inizializzati con le soluzioni calcolate da Farthest Insertion e STRONGSKIP. Per avere un quadro più completo relativamente alle

prestazioni degli algoritmi, si è supposto di avere a disposizione sia veicoli con capacità 10 per gli insieme 1 e 4, 20 per gli insiemi 2 e 5, e 40 per insiemi 3 e 6 (veicoli di tipo (a)) sia veicoli con capacità 20 per gli insieme 1 e 4, 40 per gli insiemi 2 e 5, e 80 per insiemi 3 e 6 (veicoli di tipo (b)).

Gli esperimenti sono stati eseguiti più volte facendo variare il valore minimo e massimo della lunghezza della tabu list utilizzata da ciascuno degli algoritmi TS, al fine di migliorarne le prestazioni. In particolare, negli esperimenti relativi a TS_v sono stati utilizzati i valori di soglia $\sigma = 10^{-12}$, $\sigma = \infty$ e $\sigma = 0,05$, imponendo così che, nel caso in cui tutte le migliori soluzioni non tabu verso cui effettuare il passo di ricerca locale siano peggioranti, venga rispettivamente scelta la migliore di tutti gli intorni, quella appartenente al primo intorno e la migliore dei primi S intorni (con $S = 1..4$). I risultati di seguito analizzati sono i migliori ottenuti e sono stati calcolati facendo variare la lunghezza della tabu list utilizzata da ciascun algoritmo entro i limiti indicati in tabella 4.1. I tempi di calcolo riportati sono espressi in secondi.

TS_{REL}, TS_{XWEAK}, TS_{CROSS} e TS_{3ARCX}

Le tabelle seguenti riportano, per ogni insieme di problemi, i valori medi dei risultati ottenuti risolvendo le 50 istanze con ciascuna delle implementazioni del TS che utilizzano un unico intorno; le tabelle 4.2 e 4.3 sono relative all'utilizzo di veicoli di tipo (a) mentre la 4.4 e la 4.5 sono relative all'utilizzo di veicoli di tipo (b). I valori migliori sono evidenziati in grassetto.

Dall'esame delle tabelle 4.2 e 4.4 emerge come, rispetto alla lunghezza delle soluzioni calcolate, TS_{XWEAK} sia l'algoritmo che genera le soluzioni migliori; la loro lunghezza media, rispetto a quella delle soluzioni calcolate dal corrispondente algoritmo di ricerca locale (quello che utilizza EXCHANGEWEAK), è inferiore fino al

$4,75\%$ nel caso i veicoli utilizzati siano di tipo (a) e all'$8,22\%$ nel caso in cui i veicoli siano di tipo (b).

Per quanto riguarda i veicoli utilizzati, dall'esame delle tabelle 4.3 e 4.5, si nota come TS_{XWEAK}, per come è definito EXCHANGEWEAK, non possa diminuire il numero di veicoli che caratterizza le soluzioni iniziali. TS_{REL}, TS_{CROSS} e TS_{3ARCX} permettono invece di utilizzare in media fino a due veicoli in meno nel caso in essi siano di tipo (a) e un veicolo in meno nel caso in cui essi siano di tipo (b). Il numero medio di veicoli che caratterizza le soluzioni calcolate da questi ultimi tre algoritmi è sempre minore di quello che caratterizza le soluzioni trovate dai relativi algoritmi di ricerca locale; in particolare, TS_{CROSS} e TS_{3ARCX}, rispetto ai relativi algoritmi di ricerca locale, consentono di utilizzare in media fino ad un veicolo in meno indipendentemente dal tipo.

I tempi medi di esecuzione sono stati riportati solo per dare un'idea del loro ordine di grandezza.

TS_c e TS_v

Le tabelle 4.6 e 4.9 riportano i risultati medi ottenuti eseguendo TS_{c_1}, TS_{c_2} e TS_v con i valori di soglia $\sigma = 10^{-12}$, $\sigma = \infty$ e $\sigma = 0,05$; le tabelle 4.6 e 4.7 sono relative all'utilizzo di veicoli di tipo (a) mentre la 4.8 e la 4.9 sono relative all'utilizzo di veicoli di tipo (b).

Si considerino i risultati medi riferiti agli algoritmi TS_{c_1} e TS_{c_2}. Indipendentemente dal tipo di veicoli, le prestazioni di TS_{c_2}, rispetto a quelle di TS_{c_1}, sono simili in termini di numero di veicoli utilizzati e migliori in termini di costo; in particolare TS_{c_2} genera soluzioni con un costo medio inferiore fino allo $0,62\%$ nel caso in cui i veicoli siano di tipo (a) e allo $0,34\%$ nel caso in cui i veicoli siano di tipo (b). TS_{c_2} sembrerebbe quindi da preferirsi a TS_{c_1}; se si osserva però che i tempi medi di ese-

cuzione di TS_{c_2} superano nel caso peggiore i tre minuti mentre quelli di TS_{c_1} non superano mai i trenta secondi, si intuisce come, nonostante TS_{c_2} generi soluzioni con un costo medio leggermente inferiore, sia da preferirsi l'utilizzo di TS_{c_1}.

Si confrontino ora TS_{c_1} e TS_v; indipendentemente dal tipo di veicoli utilizzati e dal valore della soglia σ sono valide le seguenti osservazioni:

- le prestazioni di TS_v, rispetto a quelle di TS_{c_1}, sono simili in termini di numero di veicoli utilizzati e migliori in termini di costo solo in pochissimi casi;
- i tempi medi di esecuzione di TS_v superano nel caso peggiore i tre minuti;

TS_{c_1} risulta essere quindi migliore anche di TS_v.

La lunghezza media delle soluzioni generate da TS_{c_1}, rispetto a quella delle soluzioni calcolate dal corrispondente algoritmo di ricerca locale (quello che utilizza l'intorno complesso), è inferiore fino al $2,14\%$ nel caso i veicoli utilizzati siano di tipo (a) e all'$4,4\%$ nel caso in cui i veicoli siano di tipo (b). Inoltre TS_{c_1}, rispetto al relativo algoritmo di ricerca locale, consente di utilizzare in media fino ad un veicolo in meno indipendentemente dal tipo.

Rispetto alle migliori soluzioni calcolabili eseguendo TS_{REL}, TS_{XWEAK}, TS_{CROSS} e TS_{3ARCX}, TS_{c_1} genera soluzioni caratterizzate da circa lo stesso numero medio di veicoli e con un costo medio inferiore di almeno il $1,66\%$ se si utilizzano veicoli di tipo (a) e di almeno il $2,53\%$ se si utilizzano veicoli di tipo (b). Inoltre il tempo medio di esecuzione di TS_{c_1}, nel caso in cui i veicoli utilizzati siano di tipo (a), è superiore a quello di TS_{XWEAK} al massimo del $52,61\%$, mentre, nel caso in cui i veicoli utilizzati siano di tipo (b), è addirittura inferiore a quello di TS_{XWEAK} fino al $31,96\%$.

In definitiva, TS_{c_1} è fra le implementazioni del TS quella migliore. Rispetto alle soluzioni calcolate da STRONGSKIP, quelle calcolate da questi algoritmi hanno un

costo medio inferiore di almeno il $15,06\%$ se si utilizzano veicoli di tipo (a) e di almeno il $19,78\%$ se si utilizzano veicoli di tipo (b); inoltre tali soluzioni prevedono di utilizzare in media fino a due veicoli in meno nel caso in cui essi siano di tipo (a) e un veicolo in meno nel caso in cui essi siano di tipo (b).

Qualità delle soluzioni

Per valutare la qualità delle migliori soluzioni ottenibili con le varie implementazioni del TS, è stato innanzitutto utilizzato come termine di paragone il *lower bound* ottenuto risolvendo con CPLEX il rilassamento lineare di ognuno dei problemi appartenenti ai sei insiemi; in particolare, per ognuno di essi è stato imposto l'utilizzo di un numero di veicoli minore o uguale a quello che caratterizza la relativa soluzione trovata da TS_{c_1}; per formalizzare i problemi è stato utilizzato il modello (1.1)-(1.8) descritto nel capitolo 1. Nelle tabelle 4.10 e 4.11 vengono riportati per ogni insieme di problemi i valori medi calcolati sulle 50 istanze del *lower bound* e dell'errore (ϵ), espresso in percentuale sul valore del *lower bound*, che caratterizza le soluzioni calcolate con TS_{c_1}. I risultati riportati mostrano che nel caso peggiore l'errore che caratterizza le soluzioni calcolate con TS_{c_1} è limitato, in media, al $17,82\%$ nel caso in cui siano utilizzati veicoli di tipo (a) e al $21,34\%$ nel caso in cui siano utilizzati veicoli di tipo (b).

In seguito CPLEX è stato utilizzato per risolvere all'ottimo le prime 10 istanze degli insiemi 1, 2 e 3 nel caso in cui siano utilizzati sia veicoli di tipo (a) sia veicoli di tipo (b); il tempo limite di esecuzione per risolvere ogni problema è stato fissato a quindici minuti ed è stato ancora imposto l'utilizzo di un numero di veicoli minore o uguale a quello che caratterizza la relativa soluzione trovata da TS_{c_1}. CPLEX è riuscito a trovare una soluzione ammissibile solo per un problema, tale soluzione prevede

l'utilizzo di un numero di veicoli pari a quello previsto dalla corrispondente soluzione trovata da TS_{c_1} ed ha un costo superiore del $5,19\%$.

		Domanda divisibile						Domanda indivisibile					
	Set	1	2	3	4	5	6	1	2	3	4	5	6
TS_{REL}	l_{min}	35	15	15	25	35	25	50	35	45	50	15	20
	l_{max}	95	100	110	140	150	150	180	180	180	140	140	150
TS_{XWEAK}	l_{min}	15	30	20	35	30	35	15	15	15	15	15	15
	l_{max}	45	45	50	75	70	70	35	35	35	40	40	40
TS_{CROSS}	l_{min}	10	16	16	18	20	20	10	10	10	12	12	12
	l_{max}	30	40	40	60	60	60	30	30	30	35	35	35
TS_{3ARCX}	l_{min}	10	10	10	10	10	10	10	10	10	10	10	10
	l_{max}	30	30	30	30	30	30	30	30	30	30	30	30
TS_{c1}	l_{1min}	25	25	40	40	40	35	25	25	25	30	30	30
	l_{1max}	135	135	135	135	135	135	80	80	80	70	70	70
	l_{2min}	20	20	30	30	30	35	10	10	10	15	15	15
	l_{2max}	65	60	65	65	65	65	40	40	40	35	35	35
TS_{c2}	l_{1min}	25	25	40	40	40	35	25	25	25	30	30	30
	l_{1max}	135	135	135	135	135	135	80	80	80	70	70	70
	l_{2min}	20	20	30	30	30	35	10	10	10	15	15	15
	l_{2max}	65	60	65	65	65	65	40	40	40	35	35	35
$TS_{v(\ [\ = 10^{-12}})$	l_{1min}	25	25	40	40	40	35	25	25	25	30	30	30
	l_{1max}	135	135	135	135	135	135	80	80	80	70	70	70
	l_{2min}	20	20	30	30	30	35	10	10	10	15	15	15
	l_{2max}	65	60	65	65	65	65	40	40	40	35	35	35
$TS_{v(\ [\ = 0,05)}$	l_{1min}	25	25	40	40	40	35	25	25	25	30	30	30
	l_{1max}	135	135	135	135	135	135	80	80	80	70	70	70
	l_{2min}	20	20	30	30	30	35	10	10	10	15	15	15
	l_{2max}	65	60	65	65	65	65	40	40	40	35	35	35
$TS_{v(\ [=\ f)}$	l_{1min}	35	15	15	25	35	25	50	35	45	50	15	20
	l_{1max}	95	100	110	140	150	150	180	180	180	140	140	150
	l_{2min}	10	16	16	18	20	20	10	10	10	12	12	12
	l_{2max}	30	40	40	60	60	60	30	30	30	35	35	35

Tabella 4.1: lunghezza minima e massima delle tabu list

		TS_{REL}		TS_{XWEAK}		TS_{CROSS}		TS_{3ARCX}	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	35,5693	30,9427	1,703	30,8348	4,368	30,9420	1,848	30,9498	39,415
2	34,7675	30,0413	1,702	29,8388	4,531	29,8607	1,806	29,9894	38,164
3	34,6180	29,5226	1,706	29,3713	5,142	29,3871	1,760	29,5005	36,728
4	65,8487	56,9992	3,489	55,3633	9,372	56,1523	6,337	56,1703	324,139
5	66,3184	56,4322	3,546	55,1418	10,572	55,4627	6,225	55,5585	318,093
6	66,6556	56,7098	3,564	54,9609	10,897	55,3437	6,269	55,4428	321,132

Tabella 4.2: veicoli di tipo (a), costo, tempo di esecuzione

	TS_{REL}	TS_{XWEAK}	TS_{CROSS}	TS_{3ARCX}
Set	h iniziale	h finale	h finale	h finale
1	18,06	17,20	18,06	17,18	17,12
2	17,44	16,72	17,44	16,62	16,64
3	16,80	16,04	16,80	16,00	15,94
4	33,48	32,06	33,48	32,12	31,90
5	33,04	31,28	33,04	31,24	31,10
6	32,66	31,48	32,66	31,42	31,38

Tabella 4.3: veicoli di tipo (a), numero di veicoli

		TS_{REL}		TS_{XWEAK}		TS_{CROSS}		TS_{3ARCX}	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	20,6681	17,6027	1,622	17,1404	12,734	18,0293	1,203	18,3428	17,122
2	19,9296	16,9060	1,527	16,5335	13,885	17,5239	1,168	17,9211	16,563
3	20,0962	16,7470	1,567	16,3088	15,647	17,1746	1,173	17,5316	17,421
4	36,7439	31,0390	3,382	29,8992	26,551	32,1277	4,356	32,3253	188,873
5	37,0301	30,4694	3,345	29,5792	30,497	31,6027	4,289	31,7143	184,735
6	37,0050	30,2435	3,367	29,2188	32,186	31,5882	4,264	31,7062	183,927

Tabella 4.4: veicoli di tipo (b), costo, tempo di esecuzione

	TS_{REL}	TS_{XWEAK}	TS_{CROSS}	TS_{3ARCX}
Set	h iniziale	h finale	h finale	h finale
1	8,92	8,56	8,92	8,48	8,58
2	8,50	8,16	8,50	8,10	8,16
3	8,22	7,92	8,19	7,94	7,96
4	16,24	15,60	16,24	15,60	15,62
5	15,76	15,14	15,76	15,12	15,08
6	15,46	15,00	15,46	14,96	14,98

Tabella 4.5: veicoli di tipo (b), numero di veicoli

		TS_{c1}		TS_{c2}		$TS_{v(\ [= 10^{-12}})$		$TS_{v(\ [= 0,05)}$		$TS_{v(\ [= f)}$	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	35,5693	30,2133	5,984	30,1650	27,180	30,2423	29,714	30,2570	28,119	30,2479	26,980
2	34,7675	29,1262	6,002	29,0755	26,693	29,1068	28,990	29,1026	27,505	29,1571	26,865
3	34,6180	28,5669	6,016	28,5651	26,129	28,6497	28,339	28,6339	27,113	28,6747	26,290
4	65,8487	54,4460	14,302	54,2444	189,859	54,4961	199,894	54,5707	178,905	54,5940	179,177
5	66,3184	53,9812	14,477	53,6470	188,897	53,9540	191,548	54,0979	176,826	53,9850	176,325
6	66,6556	53,9542	14,673	53,6562	186,872	54,0072	193,039	54,0057	179,612	54,0563	180,603

Tabella 4.6: veicoli di tipo (a), costo, tempo di esecuzione

		TS_{c1}	TS_{c2}	$TS_{v(\ [= 10^{-12}})$	$TS_{v(\ [= 0,05)}$	$TS_{v(\ [= f)}$
Set	h iniziale	h finale	h finale	h finale	h finale	h finale
1	18,06	17,20	17,10	17,18	17,20	17,22
2	17,44	16,66	16,66	16,68	16,66	16,68
3	16,80	15,98	16,00	16,04	16,04	16,04
4	33,48	32,08	31,96	32,06	32,04	32,02
5	33,04	31,42	31,02	31,24	31,24	31,18
6	32,66	31,56	31,24	31,40	31,44	31,46

Tabella 4.7: veicoli di tipo (a), numero di veicoli

		TS_{c1}		TS_{c2}		$TS_{v(\ [= 10^{-12}})$		$TS_{v(\ [= 0,05)}$		$TS_{v(\ [= f)}$	
Set	Costo iniziale	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time	Costo finale	Ex. time
1	20,6681	16,5808	9,777	16,5863	20,502	16,6619	22,632	16,6850	21,936	16,7080	21,600
2	19,9296	15,9541	10,222	15,9244	20,387	16,0425	22,728	16,0503	22,093	16,0144	21,643
3	20,0962	15,7968	10,647	15,7436	20,622	15,7816	23,052	15,8478	22,327	15,8403	22,088
4	36,7439	29,1018	23,315	29,0847	127,683	29,5695	125,704	29,6829	117,402	29,4431	116,923
5	37,0301	28,7658	24,116	28,7033	125,550	29,1398	124,702	29,2260	116,566	29,1751	115,591
6	37,0050	28,4787	24,302	28,4911	123,347	28,8555	124,932	29,0433	118,106	28,8100	117,025

Tabella 4.8: veicoli di tipo (b), costo, tempo di esecuzione

		TS_{c1}	TS_{c2}	$TS_{v(\ [= 10^{-12}})$	$TS_{v(\ [= 0,05)}$	$TS_{v(\ [= f)}$
Set	h iniziale	h finale	h finale	h finale	h finale	h finale
1	8,92	8,48	8,50	8,48	8,52	8,50
2	8,50	8,12	8,08	8,10	8,12	8,10
3	8,22	7,92	7,88	7,86	7,90	7,86
4	16,24	15,56	15,62	15,56	15,62	15,58
5	15,76	15,08	15,08	15,10	15,12	15,06
6	15,46	14,92	14,94	14,98	15,00	14,92

Tabella 4.9: veicoli di tipo (b), numero di veicoli

Set	Lower bound	(%)
1	26,2832	14,91
2	24,8605	17,24
3	24,2500	17,82
4	48,0005	13,46
5	46,8620	15,23
6	45,9761	17,36

Tabella 4.10: veicoli di tipo (a), lower bound

Set	Lower bound	(%)
1	14,0024	18,50
2	13,3258	19,84
3	13,0294	21,34
4	24,9114	16,90
5	24,3423	18,29
6	23,9148	19,12

Tabella 4.11: veicoli di tipo (b), lower bound

Conclusioni

Nella presente trattazione sono state descritte euristiche per risolvere il VRPSPD appartenenti a varie classi: gli algoritmi tour partitioning, gli algoritmi di ricerca locale e varie implementazioni del TS. Le euristiche appartenenti alle prime due classi sono state testate sia nel caso in cui la domanda dei clienti è divisibile sia nel caso in cui non lo è. A seguito della scelta di analizzare più approfonditamente le prestazioni degli algoritmi nel caso caratterizzato dalla maggiore applicabilità a problemi reali, le implementazioni del TS sono state invece testate considerando solo l'ipotesi di domanda indivisibile.

Gli esperimenti svolti hanno permesso di individuare quale o quali delle euristiche all'interno di ogni classe fosse la migliore.

Nell'ipotesi in cui la domanda sia indivisibile e che i veicoli abbiano una capacità pari al doppio della domanda massima di un cliente, viene presentato attraverso i seguenti grafici un quadro riassuntivo sulle prestazioni delle migliori euristiche di ogni classe (degli algoritmi di ricerca locale sono state considerate le prestazioni di quello che utilizza l'intorno complesso). Per ognuno dei sei insiemi di problemi su cui sono stati effettuati gli esperimenti, nei grafici 5.1, 5.2 e 5.3 vengono rispettivamente confrontati i tempi medi di esecuzione delle euristiche, ed i valori medi del costo e del numero di veicoli che caratterizzano le soluzioni da esse calcolate.

Come è possibile notare, STRONGSKIP risulta essere l'algoritmo più veloce, ma

anche quello che calcola le soluzioni peggiori sia in termini di costo sia in termini di numero di veicoli utilizzati. Rispetto a STRONGSKIP, l'algoritmo di ricerca locale che utilizza come intorno v_3 ha tempi medi di esecuzione di un ordine grandezza superiore nel caso degli insiemi 1, 2, 3 e di due ordini di grandezza nel caso degli insiemi 4, 5, 6, e migliora i valori medi del costo e del numero di veicoli che caratterizzano le soluzioni di almeno il $16,25\%$ e il $2,59\%$. Infine, rispetto all'algoritmo di ricerca locale, TS_{c_1} ha tempi medi di esecuzione di un ordine grandezza superiore nel caso degli insiemi 1, 2, 3 e fino a 3,53 volte più alti nel caso degli insiemi 4, 5, 6, e genera soluzioni con un costo medio inferiore di almeno il $3,51\%$ e caratterizzati da un numero medio di veicoli inferiore di almeno lo $0,50\%$.

Oltre alla definizione di nuove metaeuristiche, considerando i risultati ottenuti, un'altra possibile via nello sviluppo di algoritmi di approssimazione per il VRPSPD è sicuramente quella della realizzazione di euristiche di ricerca locale a due livelli sulla base dell'algoritmo realizzato da Lin e Kernighan [24] per il TSP.

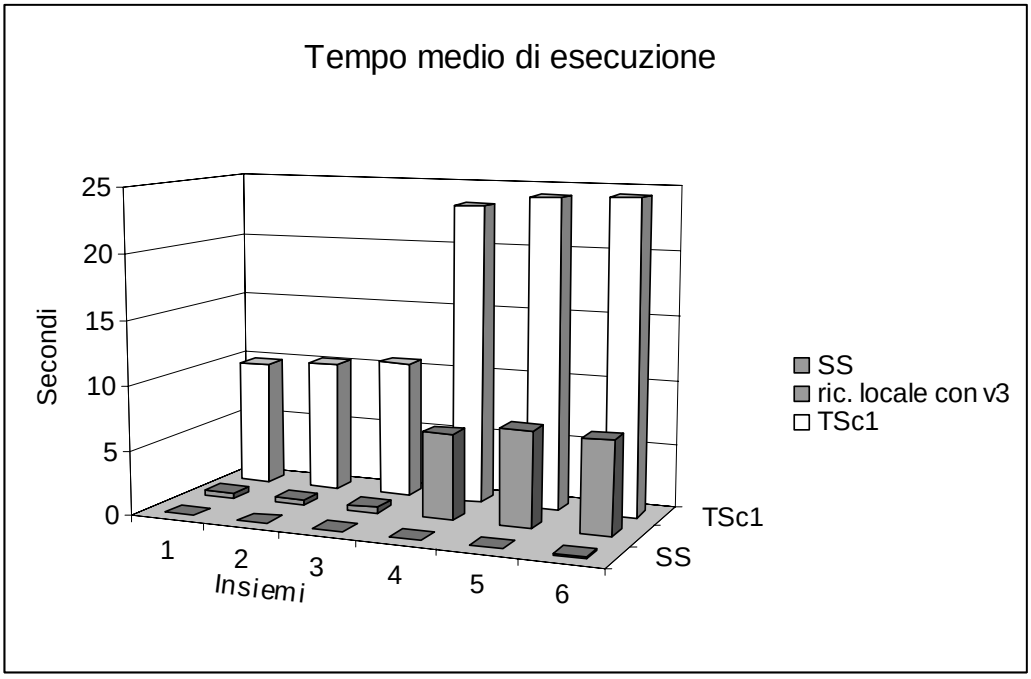

Grafico 5.1

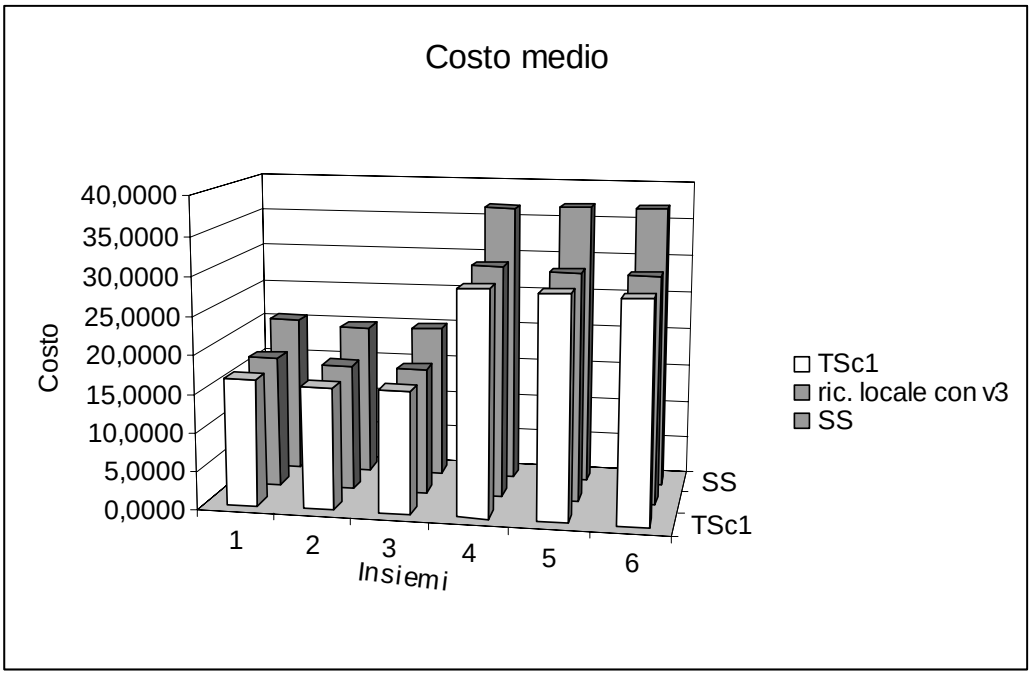

Grafico 5.2

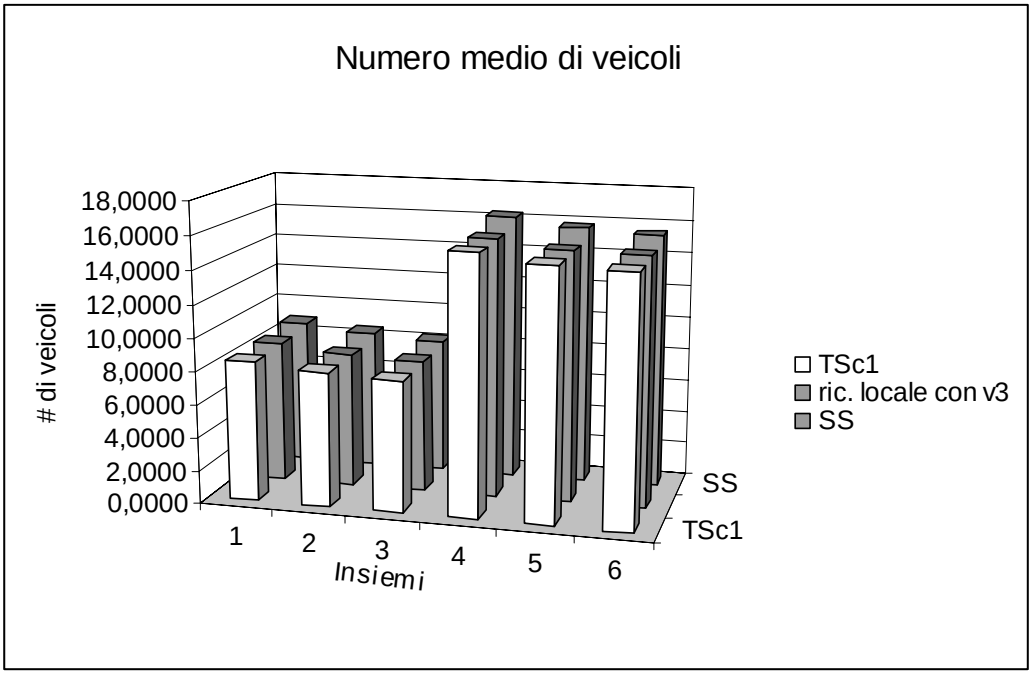
Numero medio di veicoli
18,0000
16,0000
14,0000
12,0000
10,0000
8,0000
6,0000
4,0000
2,0000
0,0000
di veicoli
1
2
3
4
5
6
Insiemi
SS
TSc1
TSc1
ric. locale con v3
SS

Grafico 5.3

Bibliografia

[1] P. Toth, D. Vigo, *The Vehicle Routing Problem.* Paolo Toth and Daniele Vigo, DEIS - University of Bologna, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, (2002), 1-26.

[2] W. M. Garvin, H. W. Crandall, J.B. John, and R. A. Spellman, *Applications of linear programming in the oil industry.* Management Science, 3, (1957), 407-430.

[3] G. Mosheiov, *Vehicle Routing with pick-up and delivery: tour partitioning heuristics.* Computer and Industrial Engineering, 34, 3, (1998), 669-684.

[4] G. Mosheiov, *The traveling salesman problem with pick-up and delivery.* European Journal of Operational Research, 72, (1994), 299-310.

[5] G. Laporte, F. Semet, *The Vehicle Routing Problem.* Paolo Toth and Daniele Vigo, DEIS - University of Bologna, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, (2002), 109-128.

[6] M. Gendreau, G. Laporte, J. Y. Potvin, *The Vehicle Routing Problem.* Paolo Toth and Daniele Vigo, DEIS - University of Bologna, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, (2002), 129-154.

[7] M. Haimovic, A. Rinnoy Kan, *Bounds and heuristics for capacitated routing problems.* Mathematics of Operations Research, 10, (1985), 527-542.

[8] K. Altinkemer, B. Gavish, *Heuristics for delivery problems with constant error guarantees.*, Transportations Science, 24, (1990), 294-297.

[9] S. Anily, G. Mosheiov, *The traveling salesman problem with delivery and backhauls.* Operations Research Letters, 16, (1994), 11-18.

[10] S. Anily, J. Bramel, *Approximation Algorithms for the Capacited Traveling Salesman Problem with Pickups and Deliveries.* Tristan III, Puerto Rico, (1998).

[11] D. J. RosenKrantz, R. E. Stearns, P. M. Lewis II, *An analysis of several heuristics for the traveling salesman problem.* Siam Journal on Computing, 6, 3[563-581], (1977).

[12] G. Righini, *The Largest Insertion algorithm for the Traveling Salesman Problem.* Note del Polo - Ricerca, 29, Polo Didattico e di Ricerca di Crema, Università di Milano, (2000).

[13] A. P. Kindervater, M. W. P. Savelsbergh, *Vehicle routing: handling edge exchanges in Local Search in Combinatorial Optimization.* Aarts E. and Lenstra J. K. eds., Wiley-Interscience Series in Discrete Mathematics and Optimization, Chichester, (1997).

[14] D. Vigo, *A heusristic algorithm for the asymmetric capacitated vehicle routing problem.* European Journal of Operational Research, 89, (1996), 108-126.

[15] P. Toth, D. Vigo, *A heusristic algorithm for the symmetric and asymmetric Vehicle Routing Problem with backhauls.* European Journal of Operational Research, 113, (1999), 528-543.

[16] P. Toth, D. Vigo, *Exact algorithms for the Vehicle Routing Problem with backhauls.* Transportation Science, 31, (1997), 60-71.

[17] M. W. P. Savelsbergh, M. Sol, *The general pickup and delivery problem.* Transportation Science, 29, (1995), 17-29.

[18] G. Righini, *Approximation algorithms for the Vehicle Routing Problem with pick-up and delivery.* Note del Polo - Ricerca, 33, Polo Didattico e di Ricerca di Crema, Università di Milano, (2000).

[19] F. Glover, É. Taillard, D. de Werra, *A user's guide to tabu search.* Annals of Operations Research, 41, (1993), 3-28.

[20] M. Gendreau, G. Laporte, D. Vigo, *Heuristics for the traveling salesman problem with pickup and delivery.* Computers and Operations Research, 26, (1999), 699-714.

[21] M. Gendreau, G. Laporte, J. Y. Potvin, *Vehicle routing: modern heuristics* in Local Search in Combinatorial Optimization, Aarts E. and Lenstra J. K. eds., Wiley-Interscience Series in Discrete Mathematics and Optimization, Chichester, (1999).

[22] M. Gendreau, A. Hertz, G. Laporte, *A Tabu Search Heuristic for the Vehicle Routing Problem.* Report CRT-777, Centre de recherche sur les transports, Université de Montréal, (1992), Management sci. to appear.

[23] I. H. Osman, *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem.* Annals of Operations Research, 41, (1993), 421-451.

[24] S. Lin, B. W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem.* Operations Research, 21, (1973), 498-516.