ALMA MATER STUDIORUM - UNIVERSITÁ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TITOLO DELL'ELABORATO

Algoritmi di ottimizzazione per il Bus Touring Problem

Elaborato in

Laboratorio Di Simulazione E Ottimizzazione

Relatore
Roberto Baldacci

Presentata da
Luca Molari

Sessione **II**
Anno Accademico **2014/2015**

Indice

Abstract

Nel campo della Ricerca Operativa e dei problemi di ottimizzazione viene presentato un problema, denominato Bus Touring Problem (BTP), che modella una problematica riguardante il carico e l'instradamento di veicoli nella presenza di di vincoli temporali e topologici sui percorsi.

Nel BTP, ci si pone il problema di stabilire una serie di rotte per la visita di punti di interesse dislocati geograficamente da parte di un insieme di comitive turistiche, ciascuna delle quali stabilisce preferenze riguardo le visite. Per gli spostamenti sono disponibili un numero limitato di mezzi di trasporto, in generale eterogenei, e di capacitá limitata. Le visite devono essere effettuate rispettando finestre temporali che indicano i periodi di apertura dei punti di interesse; per questi, inoltre, é specificato un numero massimo di visite ammesse. L'obiettivo é di organizzare il carico dei mezzi di trasporto e le rotte intraprese in modo da massimizzare la soddisfazione complessiva dei gruppi di turisti nel rispetto dei vincoli imposti.

Segue un sommario di quanto affrontato:

- La parte 1 é dedicata alla descrizione del Bus Touring Problem;
- Nella parte 2 viene presentato un algoritmo euristico basato su Tabu Search per la risoluzione del BTP, e per esso un'implementazione esemplificativa;
- Nella parte 3 vengono presentati gli esperimenti effettuati mettendo appunto i parametri dell'algoritmo su un insieme di problemi di benchmark.
- Nella parte 4 vengono presentate le conclusioni, con considerazioni ed indicazioni di sviluppo futuro.
- In appendice (parte 5) vengono presentati i risultati estesi dell'esecuzione dell'algoritmo su tutte le istanze di benchmark.

Parte 1

Bus Touring Problem

CAPITOLO 1

Introduzione

Il Bus Touring Problem (BTP) coinvolge un insieme di gruppi di turisti, un insieme di mezzi di trasporto, ed un insieme di attrazioni turistiche - dislocate geograficamente - che é possibile visitare.

Per ciascuna attrazione turistica, denominata POI (Point of Interest) , é definito un intervallo temporale entro il quale la visita da parte delle comitive di turisti puó avvenire. Ciascun gruppo di turisti stabilisce le proprie preferenze riguardo la visita dei punti di interesse, attribuendo a ciascuno di essi un punteggio numerico che indica il grado soddisfazione che si raggiunge nel caso in cui si riesca a visitare il particolare POI. Per effettuare gli spostamenti fra i punti di interesse si ha a disposizione un numero limitato di mezzi di trasporto, ciascuno di essi caratterizzato da una capacitá massima di carico; si assume inoltre che tutti i mezzi di trasporto abbiano la stessa velocitá di percorrenza.

POI

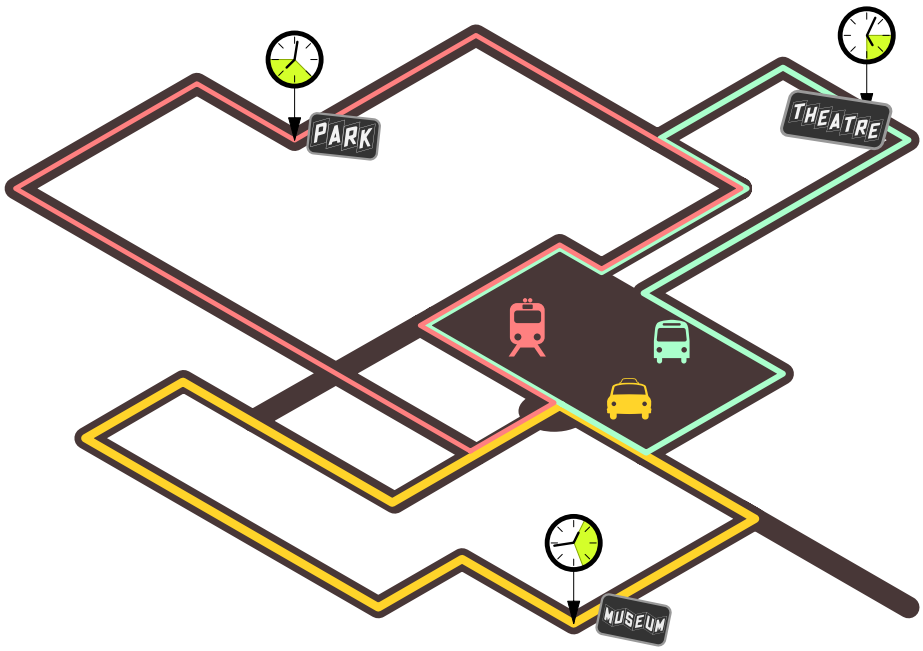
Turisti

Bus

L'obiettivo é quello di organizzare le visite delle comitive alle attrazioni turistiche attraverso una serie di percorsi chiusi (tour), che iniziano e terminano in corrispondenza di un particolare punto geografico denominato deposito (Depot), e che si concludono entro un orario massimo stabilito. La definizione di un Tour, quindi, comprende:

Tour

- La scelta di un particolare mezzo di trasporto tra quelli disponibili;
- La scelta dei gruppi di turisti assegnati al Tour;
- La selezione e l'ordine di visita dei POI visitati dal Tour, ovvero la Route intrapresa.



1. Motivazione

I problemi di ottimizzazione sono generalmente enunciati utilizzando uno scenario applicativo emblematico, per agevolare il ragionamento sul determinato problema e rendere la trattazione piú comprensibile. Nel nostro caso ci si riferisce ad uno scenario turistico (Bus Touring), perché la trasposizione del problema astratto in esso risulta particolarmente agevole.

Tuttavia, il problema qui presentato consente di modellare tutta una serie di differenti situazioni reali al di lá dello scenario in cui viene enunciato. Ciascuna entitá coinvolta nel problema, infatti, puó essere considerata come un'astrazione che cattura un concetto generico.

In questo senso, l'astrazione di POI concettualizza l'accesso ad un profitto, la distanza tra i POI rappresenta il dispendio di risorse necessario ad accedere a tale profitto, e questa, insieme al concetto di finestra temporale, porta vincoli sulla sequenzializzazione degli accessi. Anche il concetto di finestra temporale in realtá é una specificitá introdotta per convenienza. Il concetto piú astratto che rappresenta é in realtá un vincolo di inclusione che opera in una dimensione omogenea - o rapportabile - alla dimensione coinvolta nella sequenzializzazione degli accessi ai profitti (nel nostro caso pratico la dimensione é temporale).

I gruppi di turisti sono coinvolti nella determinazione dell'entitá del profitto raccolto, e anche per essi é presente una componente che controbilancia tale profitto, ovvero l'occupazione di capacitá disponibile.

I Bus rappresentano una risorsa scarsa, necessaria per accedere ai profitti, e che stabilisce il limite sul profitto che é possibile raccogliere contestualmente alla visita di un POI.

In un'ottica cosí astratta, é semplice identificare alcune classi di situazioni reali che possono essere rappresentate dalla formulazione BTP, ad esempio problemi sulla distribuzione di merci e servizi, scenari che coinvolgono lo scheduling e la pianificazione di attivitá, determinazione di schemi di investimenti, etc.

CAPITOLO 2

Descrizione del problema

Il Bus Touring Problem é definito su un digrafo completo $G = (V, A)$, dove V é l'insieme dei vertici ed A l'insieme degli archi. L'insieme dei vertici é composto dal vertice 0 e da un insieme P che include dei punti di interesse (POI, Points of Interest), o, in altre parole, punti della cittá che possono potenzialmente suscitare interesse per un turista. I POI possono rappresentare monumenti, musei, gallerie, parchi a tema, ecc. Il vertice 0 rappresenta il punto dal quale i percorsi di visita partono e terminano. Per ogni arco $(i_1, i_2) \in A, i_1, i_2 \in V, i_1 \neq i_2$ é associato un tempo di percorrenza $\hat{t}_{i_1 i_2}$.

Per ogni POI $i \in P$, esiste una finestra temporale (TW, time-window) $[e_i, l_i]$ che definisce l'intervallo entro il quale l'inizio di una visita puó iniziare. Inoltre, viene considerata una durata temporale s_i che rappresenta il tempo necessario perché una visita del POI venga portata a termine. I tempi di percorrenza $\hat{t}_{i_1 i_2}$ possono essere modificati per tenere in considerazione questo tempo di visita s_i, dando cosí origine ad una matrice modificata dei tempi $r_{i_1 i_2}$. É importante notare che se un Bus raggiunge il POI i in un istante temporale che precede e_i, i passeggeri sono costretti ad attendere l'istante di apertura e_i di questo particolare punto di interesse.

Al punto di smistamento 0, si trova un insieme di Bus B. Per ogni Bus $k \in B$, risulta associata una capacitá Q_k, che indica il massimo numero di passeggeri che esso puó contenere. Inoltre, anche per il vertice 0 risulta definita una finestra temporale $[e_0, l_0]$, corrispondente all'intervallo per cui é ammesso il rientro da un percorso di visite.

Viene considerato un insieme di comitive di turisti T. Ogni gruppo di turisti $j \in T$ consiste in un gruppo di q_j persone che intende viaggiare insieme (quindi nello stesso Bus). Il gruppo di turisti puó quindi modellare una famiglia, un gruppo di amici, o un turista che viaggia da solo. Per esprimere le proprie preferenze, ogni gruppo di turisti $j \in T$ dichiara quanto é importante la visita di ogni POI. In questo modo, per ogni gruppo $j \in T$, ed ogni POI $i \in P$, risulta associato un profitto $p_{ji} \geq 0$, che indica la propensione del gruppo di turisti a visitare il punto di interesse. Ovviamente ne consegue che se un gruppo j viene assegnato ad un Bus k che visita il POI i, viene realizzato il relativo profitto p_{ji}.

Si assume che il tempo di visita sia indipendente dal numero di gruppi e dal numero di persone. Inoltre, ciascun POI puó essere visitato al piú da K Bus.

Obiettivo

Lo scopo del problema é di decidere congiuntamente l'assegnazione dei gruppi di turisti ai Bus, e, per ogni Bus, il percorso da intraprendere (tour, o Route). In termini analitici, l'obiettivo é di decidere riguardo il carico dei Bus ed il relativo routing rispettando i seguenti vincoli:

(1) Assegnazione gruppi di turisti ↔ Bus:
   (a) Ogni gruppo di turisti deve essere assegnato ad uno ed un solo Bus.
   (b) Il numero totale di passeggeri di un Bus non deve eccedere la relativa capacitá.

(2) Bus routing:
   (a) Ogni Route deve iniziare al vertice 0 e terminare al vertice 0.
   (b) Ogni Bus puó visitare una POI al massimo una volta.

(c) Ogni POI puó essere visitato al massimo un numero K di volte.
(d) Ogni Bus deve raggiungere i POI prima del corrispondente istante di chiusura. É ammesso arrivare prima dell'istante di apertura, ma in questo caso occorre attendere tale istante per poter iniziare la visita.

In termini di funzione obiettivo, lo scopo principale é quello di massimizzare il profitto totale dato dalla somma dei profitti dei singoli gruppi di turisti.

CAPITOLO 3

Formulazione matematica

Presentiamo qui due formulazioni matematiche per il Bus Touring Problem.

Una Route o Bus Tour $R = \left(i_0 = 0, i_1, \dots, i_n = 0\right)$ é un circuito semplice (ovvero senza ripetizione di archi, e di conseguenza senza ripetizione di vertici) in G che parte dal Depot all'istante temporale e_0 e termina al Depot prima dell'istante temporale l_0. Questa Route R visita un sottoinsieme di POI $V(R) = \left\{i_1, \dots, i_n\right\}$.

Una Route R é ammissibile se soddisfa il seguente vincolo:

time window: il tempo di arrivo $\tau_{i_h}, 1 \leq h \leq n$, a ciascun vertice $i_h \in V(R)$ deve rispettare la finestra temporale rappresentata dall'equazione

$$\max\left\{\tau_{i_{h-1}} + t_{i_{h-1}i_h}, e_{i_h}\right\} \leq \tau_{i_h} \leq l_{i_h}$$

,dove i_{h-1} indica il vertice visitato immediatamente prima di i_h nella Route R (e $\tau_0 = e_o$).

Sia $\mathscr{R}$ il set contenente gli indici di tutte le Route ammissibili, e $a_{il} \in \{0, 1\}$ un coefficiente uguale a 1 se e solo se la Route $\ell \in \mathscr{R}$ visita il POI $i \in P$. Nelle seguenti formulazioni la notazione R_ℓ denota la sequenza ordinata di vertici visitati dalla Route $\ell \in \mathscr{R}$.

1. Formulazione *F1*

Siano:

$x_\ell^k \in (0,1)$**:** variabile binaria con valore 1 se e solo se la Route $\ell \in \mathscr{R}$ é in soluzione per il bus $k \in B$;

$y_{ij}^k \in (0,1)$**:** variabile binaria con valore 1 se e se il POI $i \in P$ é visitato dal gruppo di turisti $j \in T$ utilizzando il bus $k \in K$;

$z_j^k \in (0,1)$**:** variabile binaria uguale a 1 se e solo se il gruppo di turisti $j \in T$ utilizza il bus $k \in K$.

La formulazione *F1* del BTP specifica:

$$\max \phi = \sum_{i \in P} \sum_{j \in T} \sum_{k \in B} p_{ji} y_{ij}^k \tag{3.1}$$

$$\text{s.t.} \sum_{\ell \in \mathscr{R}} x_\ell^k \leq 1, \qquad \forall k \in B \tag{3.2}$$

$$\sum_{\ell \in \mathscr{R}} a_{i\ell} x_\ell^k \leq 1 - z_j^k + y_{ij}^k, \qquad \forall i \in P, \forall j \in T, \forall k \in B \tag{3.3}$$

$$\sum_{k \in B} \sum_{\ell \in \mathscr{R}} x_\ell^k \leq K, \qquad \forall i \in P \tag{3.4}$$

$$y_{ij}^k \leq \sum_{\ell \in \mathscr{R}} a_{i\ell} x_\ell^k, \qquad \forall i \in P, \forall j \in T, \forall k \in B \tag{3.5}$$

$$y_{ij}^k \leq z_j^k, \qquad \forall i \in P, \forall j \in T, \forall k \in B \tag{3.6}$$

$$\sum_{j \in T} q_j z_j^k \leq Q_k, \qquad \forall k \in B \tag{3.7}$$

$$\sum_{k \in B} z_j^k = 1, \qquad \forall j \in T \tag{3.8}$$

$$x_\ell^k \in \{0,1\}, \qquad \forall \ell \in \mathscr{R}, \forall k \in B \tag{3.9}$$

$$y_{ij}^k \in \{0,1\}, \qquad \forall i \in P, \forall j \in T, \forall k \in B \tag{3.10}$$

$$z_j^k \in \{0,1\}, \qquad \forall j \in T, \forall k \in B \tag{3.11}$$

Dove:

- La funzione obiettivo (3.1) massimizza il profitto totale.
- I vincoli (3.2) impongono che per ogni Bus sia selezionata al massimo una Route.
- I vincoli (3.3) assicurano che se un gruppo j é assegnato al Bus k visitando il POI i, allora $y_{ij}^k = 1$.
- I vincoli (3.4) fanno si che un POI i possa essere visitato complessivamente un massimo di K volte.
- I vincoli (3.5) e (3.6) assicurano che un POI i é visitato dal gruppo di turisti j usando il Bus k solo se una Route che visita i é assegnata al Bus k e il gruppo di turisti j é assegnato al Bus k.
- I vincoli (3.7) garantiscono che il numero di turisti assegnati ad un Bus k non ecceda la relativa capacitá Q_k.
- I vincoli (3.8) indicano che ogni gruppo di turisti j debba essere assegnato ad esattamente 1 Bus.
- Infine, i vincoli (3.9), (3.10) e (3.11) impongono alle variabili x, y e z di essere binarie.

2. Formulazione *F2*

Siano:

W_k: il set di tutti i possibili carichi di passeggeri per un generico Bus $k \in B$;

x^k_{wl}: variabile binaria con valore 1 se e solo se la Route $\ell \in \mathcal{R}$ é in soluzione per il Bus $k \in B$ con assegnazione di carico $w \in W_k$;

$y^k_{j\ell}$: variabile binaria con valore 1 se e solo se il gruppo di turisti $j \in T$ é assegnato alla Route $\ell \in \mathcal{R}$ del Bus $k \in B$.

La formulazione *F2* del BTP specifica:

$$\max \phi = \sum_{j\in T}\sum_{k\in B}\sum_{\ell\in\mathcal{R}} c_{j\ell} y^k_{j\ell} \tag{3.12}$$

$$\text{s.t.} \sum_{k\in B}\sum_{\ell\in\mathcal{R}} y^k_{j\ell} = 1, \qquad \forall j \in T \tag{3.13}$$

$$\sum_{\ell\in\mathcal{R}}\sum_{w\in W_k} x^k_{w\ell} = 1, \qquad \forall k \in B \tag{3.14}$$

$$\sum_{k\in B}\sum_{\ell\in\mathcal{R}}\sum_{w\in W_k} a_{i\ell} x^k_{w\ell} \le K, \qquad \forall i \in P \tag{3.15}$$

$$\sum_{j\in T}\sum_{\ell\in\mathcal{R}} q_j y^k_{j\ell} = w \sum_{\ell\in\mathcal{R}} x^k_{w\ell}, \qquad \forall w \in W_k, \forall k \in B \tag{3.16}$$

$$\sum_{j\in T} y^k_{j\ell} \le |T| \sum_{w\in W_k} x^k_{w\ell}, \qquad \forall k \in B, \forall \ell \in \mathcal{R} \tag{3.17}$$

$$x^k_{w\ell} \in \{0,1\}, \qquad \forall \ell \in \mathcal{R}, \forall w \in W_k \forall k \in B \tag{3.18}$$

$$y^k_{j\ell} \in \{0,1\}, \qquad \forall \ell \in \mathcal{R}, \forall k \in B, \forall j \in T \tag{3.19}$$

Dove:

- La funzione obiettivo (3.12) massimizza il profitto totale.
- I vincoli (3.13) impongono che ogni gruppo di turisti sia assegnato ad un unica Route.
- I vincoli (3.14) impongono che ogni Bus intraprenda al massimo un solo tour.
- I vincoli (3.15) impongono che un POI $i \in P$ possa essere visitato da un numero massimo K di Bus.
- I vincoli (3.16) e (3.17) collegano le variabili x ed y.
- Infine, i vincoli (3.18) e (3.19) impongono alle variabili x ed y di essere binarie.

3. Relazioni con altri problemi

Il Bus Touring Problem si colloca tra quell'insieme di problemi che vengono formulati combinando modelli provenienti da problemi di carico (es. Knapsack, varianti di bin-packing, etc.), con modelli provenienti da problemi di instradamento di veicoli (VRP, TSP, etc.).

Il BTP generalizza un particolare insieme di formulazioni che si fondano sull'Orienteering Problem (OP) [CGW96a]. Nell'OP é definito un grafo di routing[1] con un profitto associato a ciascun vertice. L'obiettivo é di determinare un cammino di durata massima fissata che raccolga il massimo profitto possibile. OP

Un'estensione naturale di questo problema si ottiene portando il numero di Route da determinare simultaneamente da 1 a P: questa generalizzazione prende il nome di Team Orienteering Problem (TOP) [CGW96b]. TOP

L'introduzione di finestre temporali a questi due problemi ha dato origine alle varianti OPTW e TOPTW. Questa serie di vincoli aggiuntivi aumenta notevolmente la difficoltá (T)OPTW

[1] Grafo con archi pesati che rappresentano tempi di percorrenza

di risoluzione e rende necessarie modifiche agli algoritmi risolutivi impiegati nelle varianti classiche: ad esempio, l'efficacia della ben nota mossa 2-OPT (scambio intra-route di archi), impiegata da molti risolutori euristici, risulta notevolmente ridotta in presenza di vincoli di time-window [VSVO11][2].

Un'indagine approfondita sulle varianti dell'Orienteering Problem si puó trovare in [VSVO11].

BTP

Il Bus Touring Problem generalizza il problema TOPTW generalizzando il concetto di Team in quello di Bus: mentre per la formulazione TOPTW il profitto ricavato dalla visita di un vertice é costante, per la formulazione BTP esso dipende dalla composizione del "team" (Bus) che lo visita, ovvero dall'assegnazione dei gruppi di turisti al Bus. Questa assegnazione non puó avvenire in maniera arbitraria, ma é regolata da vincoli di capacitá ed esclusivitá (simili a quelli del Multiple Knapsack Problem), ed esaustivitá (in quanto tutti i gruppi di turisti devono essere assegnati).

4. Formulazione BTP di (T)OPTW

Si mostra come le formulazioni OPTW e TOPTW siano contenute nella formulazione BTP, e ne rappresentino, quindi, una degenerazione.

4.1. OPTW in BTP. L'Orienteering Problem con Time Window (OPTW) é definito in un digrafo completo $G' = \left(V', A'\right)$, dove V' rappresenta l'insieme dei vertici ed A' l'insieme degli archi. Il vertice 0 rappresenta il deposito (Depot).

Siano $\bar{p}_i \geq 0, \bar{p}_0 = 0$ i profitti associati ai vertici $i \in V'$, $\bar{t}_{ij} \geq 0$ i tempi di percorrenza associati a ciascun arco $(i, j) \in A'$, e $\bar{t}_0$ il tempo massimo consentito di percorrenza del veicolo. Sia inoltre $\left[\bar{e}_i, \bar{l}_i\right]$ la finestra temporale associata a ciascun vertice $i \in V' \setminus \{0\}$.

L'OPTW consiste nel trovare un percorso per il veicolo, visitando ciascun vertice in $V' \setminus \{0\}$ al massimo una volta e rispettando per ciascuno la corrispondente finestra temporale, impiegando un tempo non superiore $\bar{t}_0$, e collezionando il profitto massimo.

Una generica istanza OPTW puó essere convertita in un'istanza equivalente BTP secondo la seguente procedura:

(1) Definire il grafo BTP $G = (V, A)$ impostando:
   (a) $V = V'$,
   (b) $A = A'$,
   (c) $P = V' \setminus \{0\}$,
   (d) $t_{ij} = \bar{t}_{ij}, \forall (i, j) \in A$;
(2) Definire $|B| = 1$, $Q_1 = 1$, $K = 1$;
(3) Definire $|T| = 1$, $q_1 = 1$, $p_{1i} = \bar{p}_i, \forall i \in V' \setminus \{0\}$;
(4) Definire $\left[e_0, l_0\right] = \left[0, \bar{t}_0\right]$, $\left[e_i, l_i\right] = \left[\bar{e}_i, \bar{l}_i\right], \forall i \in V' \setminus \{0\}$.

Questo consiste sostanzialmente nel considerare un BTP con un solo Bus di capacitá unitaria, un solo gruppo di turisti di dimensione unitaria, e tempi di servizio s_i identicamente nulli.

4.2. TOPTW in BTP. Il Team Orienteering Problem con Time Windows é un OPTW dove l'obiettivo é quello di determinare m Route che massimizzano il profitto totale.

Una generica istanza TOPTW puó essere convertita in un'istanza BTP seguendo la procedura 4.1 per quanto riguarda il grafo e le time-window. La differenza consiste nella definizione dei Bus e dei gruppi di turisti:

(1) Definire $|B| = m$, $Q_k = 1, \forall k \in B$, $K = 1$;
(2) Definire $|T| = m$, $q_j = 1, \forall j \in T$, $p_{ji} = \bar{p}_i, \forall j \in T, \forall i \in V' \setminus \{0\}$.

Questo differisce dalla traduzione OPTW nel fatto di considerare m Bus omogenei di capacitúnitaria, m gruppi di turisti di dimensione unitaria, e stabilire per ciascuno di essi le stesse preferenze per i POI, replicando per ciascuno di essi il vettore dei profitti TOPTW.

[2] Pieter Vansteenwegen, Wouter Souffriau e Dirk Van Oudheusden. «The orienteering problem: A survey». In: *European Journal of Operational Research* 209 (2011), pp. 1–10.

Parte 2

Algoritmo risolutivo

CAPITOLO 4

Meta-euristiche e Tabu Search

Nel campo dell'ottimizzazione esistono due grandi famiglie di algoritmi risolutivi:

- Gli algoritmi detti *esatti* compiono una risoluzione per via analitica, e garantiscono di terminare restituendo una soluzione ottima;
- Gli algoritmi detti *euristici* operano applicando una risoluzione per via approssimata basandosi appunto su un'euristica, ovvero sullo sfruttamento della conoscenza - generalmente incompleta - che si ha riguardo la struttura della classe di problemi interessati.

Gli algoritmi euristici offrono quindi garanzie di tipo best-effort, ovvero cercano di reperire la soluzione migliore possibile nel rispetto dei vincoli sulle risorse che vengono messe a disposizione.

In virtú delle garanzie che offrono gli algoritmi esatti, sorge spontanea la domanda sul perché non vengano sempre impiegati questi ultimi. Innanzitutto, é possibile che per il dato problema non esista un algoritmo esatto. In secondo luogo, l'algoritmo esatto puó essere computazionalmente proibitivo anche per problemi di piccole dimensioni (problemi NP-Hard); questo, purtroppo, avviene per la maggior parte dei problemi di interesse pratico.

La ricerca nel campo degli algoritmi euristici procede nel tentativo di colmare questo divario: analizzando il problema specifico che si vuole risolvere si cerca di rendere la risoluzione trattabile attraverso una serie di semplificazioni e compromessi, che in generale pregiudica la garanzia sull'ottimalitá della soluzione trovata.

É quindi chiaro che un algoritmo euristico sia sviluppato ad-hoc su una particolare formulazione di problema. Tuttavia, nel corso dello sviluppo in questo campo sono emerse tecniche e strategie di alto livello che possono effettivamente essere impiegate indipendentemente dal problema specifico, denominate *meta-euristiche* . Una meta-euristica guida il processo risolutivo, che assume quindi la forma di una ricerca nello spazio delle soluzioni. La strategia di ricerca é sviluppata cercando di rendere efficiente questo processo, favorendo l'esplorazione di soluzioni vicine all'ottimalitá.

Meta-Euristiche

Dato che le meta-euristiche nascono per propria natura come approcci sperimentali, esse vengono sviluppate in maniera empirica e spesso sono ispirate a processi biologici o fisici, o a modelli di sistemi complessi, e spesso coinvolgono fenomeni stocastici. Le meta-euristiche ideate sono ad oggi numerose, e molte di esse condividono concetti comuni. Parte della letteratura in questo campo si é occupata della loro analisi e classificazione[1][2].

Uno dei principali fattori di classificazione per le meta-euristiche riguarda la maniera con cui viene interpretato il concetto di soluzione.

Gli approcci basati sul concetto di popolazione considerano un gruppo di soluzioni candidate; questo gruppo viene generato e sottoposto a modifiche nel tentativo di migliorarne la qualitá. Seguono questo approccio gli algoritmi di tipo ant colony, particle swarm, algoritmi genetici e di tipo evoluzionistico.

Un'altra classe di algoritmi invece considera il miglioramento di un'unica soluzione, nella maggior parte dei casi attraverso l'esplorazione di un suo vicinato; i diversi approcci

[1] A. Roli e C. Blum. «Metaheuristics in combinatorial optimization: Overview and conceptual comparison». In: *ACM Computing Surveys* (2003), pp. 268–308.

[2] L. Bianchi et al. «A survey on metaheuristics for stochastic combinatorial optimization». In: *Natural Computing: an international journal* (2009), pp. 238–287.

secondo i quali viene definito ed esplorato danno origine ad una molteplicitá di algoritmi differenti. Uno di questi algoritmi é detto tabu-search, per la caratteristica distintiva con cui parte del vicinato viene temporaneamente escluso dal processo di esplorazione. Questo é l'algoritmo con cui si intende sperimentare la risoluzione del Bus Touring Problem.

1. Tabu Search

Sebbene i concetti alla base del Tabu Search (TS) abbiano origine negli anni '70, la sua formulazione moderna del é stata introdotta e sviluppata da Glover[3][4][5].

Il TS consiste nel guidare il processo di ricerca locale nello spazio delle soluzioni cercando di evitare quei problemi principali tipici dell'ottimizzazione di problemi complessi: la condizione stallo sugli estremanti locali della funzione obiettivo (che in genere é fortemente non lineare), e l'insorgenza di cicli nella visita.

Memoria

Questi problemi vengono mitigati principalmente attraverso il concetto di *memoria parziale*: il processo di esplorazione viene arricchito da uno stato che aiuta a riconoscere ed evitare fenomeni di stallo o ciclicitá nella visita.

Sebbene esistano moltissime varianti implementative di questo concetto, nella maggior parte dei casi la memoria viene differenziata sulla base della scala temporale di persistenza: é quindi comune dotare il processo di esplorazione di una gerarchia (spesso eterogenea) di memoria. All'apice di questa gerarchia c'é una memoria di breve termine, nella quale l'elevato dettaglio informativo é bilanciato da una ridotta persistenza temporale. Nei livelli inferiori il dettaglio diminuisce progressivamente, mentre il grado di persistenza aumenta. Questo bilanciamento permette di mantenere il consumo di memoria entro livelli computazionalmente accettabili.

Short-Term Memory: Tabu List

La memoria di breve termine impiegata nel TS é la cosiddetta tabu-list , il cui meccanismo di funzionamento é talmente emblematico da conferire il nome all'algoritmo. La tabu-list individua una serie di soluzioni da considerare tabu: soluzioni che l'algoritmo rifiuta categoricamente di esplorare.

Per codificare le soluzioni all'interno della tabu-list viene generalmente utilizzata una delle seguenti modalitá, ciascuna con differenti pregi e difetti:

- Codifica completa della soluzione;
- Specifica implicita della soluzione attraverso tratti caratteristici (approccio a "DNA");
- Specifica implicita attraverso le mosse che hanno dato origine alla soluzione (approccio costruttivo).

Esplorazione non necessariamente migliorante

La tabu-list é giá sufficiente ad evitare che la ricerca si arresti su eventuali estremanti locali, se si considera la seguente strategia di esplorazione del vicinato :

(1) Ad un generico passo di esplorazione, si genera un vicinato per la soluzione corrente (variabile a seconda dell'implementazione);
(2) Si scarta da tale vicinato l'insieme delle soluzioni individuate dallo stato corrente della tabu-list;
(3) Si seleziona dal vicinato la soluzione migliore disponibile.

Il fattore importante é che ci si sposta sulla soluzione migliore disponibile nel vicinato anche se questa non é migliore della soluzione ottima corrente. Questo fattore impedisce l'arresto su un estremante locale. Considerando che la ricerca non si svolge unicamente in direzioni strettamente miglioranti, é ovviamente necessario mantenere in memoria la soluzione migliore finora trovata dall'algoritmo. Questa memoria é collocabile nella gerarchia sopra introdotta come memoria di persistenza massima: essa infatti deve perdurare per l'intero processo di risoluzione.

Long-Term Memory: Miglior Soluzione

[3] Fred W. Glover. «Future Paths for Integer Programming and Links to Artificial Intelligence». In: *Computers and Operations Research* 13 (1986), pp. 533–549.

[4] Fred W. Glover. «Tabu Search - Part I». in: *ORSA Journal on Computing* 1 (1989), pp. 190–216.

[5] Fred W. Glover. «Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges». In: *Interfaces in Computer Science and Operations Research*. 1996, pp. 1–75.

Riassumendo, una variante di base dell'algoritmo Tabu Search puó essere espressa nella maniera seguente:

Algorithm 1 Tabu Search: variante di base con tabu-list (problema di max).

1: Generare una soluzione iniziale $i \in \mathbb{S}$.
2: $i^* \leftarrow i$, $k \leftarrow 0$
3: $\mathbb{T} \leftarrow \emptyset$, SETTABU($\mathbb{T}, i, k$) ▷ Init della tabu-list
4: **repeat**
5: CLEARTABU($\mathbb{T}, k$) ▷ Eliminare soluzioni vecchie dalla tabu-list
6: Generare un vicinato $N(i,k) \subseteq \mathbb{S}$.
7: $j^* = j \mid f(j) > f\left(j_1\right), \forall j, j_1 \in N(i,k)$ ▷ Criterio di scelta della soluzione
8: $i \leftarrow j$ ▷ Spostamento sulla soluzione j
9: **if** $f(i) > f(i^*)$ **then**
10: $i^* \leftarrow i$ ▷ Nuova soluzione ottima
11: **end if**
12: SETTABU($\mathbb{T}, i, k$) ▷ Aggiornamento tabu-list
13: $k \leftarrow k + 1$
14: **until** condizione di stop

Alcune condizioni di stop comuni possono essere:

- Generazione di un vicinato vuoto $N(i,k) \equiv \emptyset$;
- Raggiungimento di un massimo numero di iterazioni $k \equiv k_{max}$;
- Sono oltrepassato un numero massimo di iterazioni dall'ultimo miglioramento di i^*.

1.1. Intensificazione e diversificazione. Una variante al processo di selezione della soluzione corrente a partire dal vicinato consiste nell'impiegare una *funzione di score* "modificata" $\bar{f}()$ al posto della funzione obiettivo $f()$. Questo permette di introdurre aspetti di *intensificazione e diversificazione* della ricerca.

Quando nella funzione di score vengono introdotti termini riguardanti variabili di stato - generalmente a medio termine - relativi al processo risolutivo, si rende possibile la modulazione reattiva della dinamica di esplorazione. Mid-Term Memory

Questa modulazione comunemente consiste nell'intensificare la visita qualora l'euristica impiegata realizzi di trovarsi in una regione buona dello spazio delle soluzioni, facendo si che $\bar{f}$ prediliga soluzioni nel vicinato di quella corrente. Allo stesso modo, é possibile diversificare la visita qualora si identifichi euristicamente una regione corrente particolarmente infruttuosa.

In una variante di Tabu Search, quest'ultimo processo di diversificazione viene portato all'estremo, stabilendo che - sotto determinate condizioni - si accetti di uscire temporaneamente dalla regione ammissibile, nel (talvolta disperato) tentativo di rientrare in un'area favorevole all'esplorazione. Uscita e rientro dalla regione ammissibile

Questa variante rappresenta un incremento notevole del potere risolutivo. Si consideri che l'esplorazione avviene mediante una serie di passi, ciascuno dei quali consiste nella generazione di un vicinato e nell'esplorazione di una soluzione ad esso appartenente. Risulta ovvio che un'eventuale convergenza del risolutore euristico alla soluzione ottima esatta puó avvenire solo nel caso in cui questa sia raggiungibile; in altri termini, se esiste per il risolutore una sequenza contigua e percorribile di vicinati, dei quali il primo contiene la soluzione iniziale e l'ultimo quella ottima. Una sola "interruzione" pregiudica completamente questa possibilitá: sotto questo punto di vista, la possibilitá di uscire dalla regione ammissibile permette di aumentare il fattore connettivo della regione effettivamente esplorabile. Raggiungibilitá

Gli stadi Tabu Search impiegati nel nostro risolutore sperimentale sono stati modellati sulla base di quanto qui presentato. Per dettagli implementativi a riguardo, si rimanda a 6.

CAPITOLO 5

Presentazione dell'algoritmo

Il framework é stato progettato principalmente sulla base del tipo di risolutore che si intendeva realizzare, tenendo peró in considerazione dell'esigenza di poter sperimentare varianti ed intervenire sulla struttura del risolutore in maniera relativamente agevole. Si é cercato quindi di dotare il framework di una certa modularitá.

Presentiamo in questo capitolo i concetti principali su cui esso si fonda.

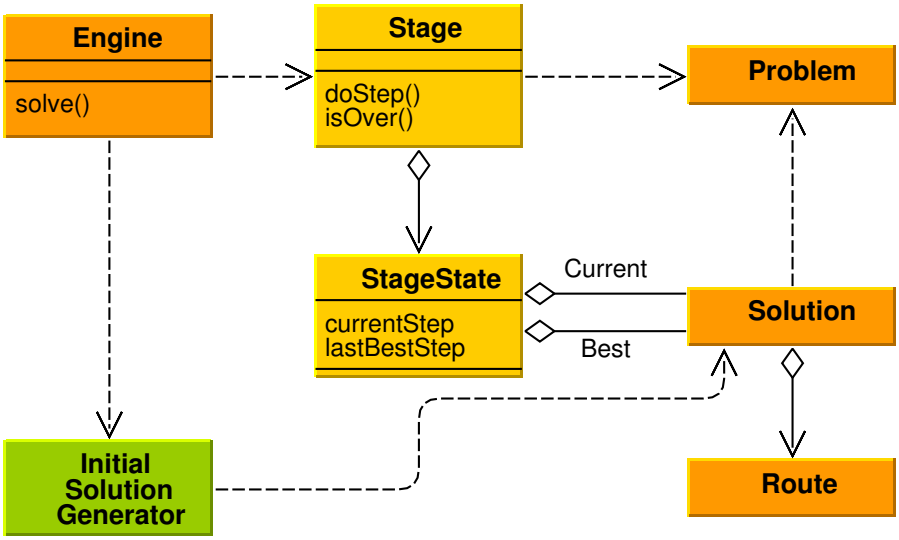


FIGURA 1. Organizzazione concettuale del framework.

1. Stadio (Stage)

L'astrazione principale che permette di definire un risolutore all'interno del framework é quella di stadio risolutivo, rappresentato dall'interfaccia Stage. Uno stadio é un componente che implementa una specifica politica risolutiva definendo le operazioni da svolgere in un passo elementare di risoluzione, detto step: queste operazioni determinano in generale un cambiamento dello stato interno del risolutore.

Uno stadio specifica inoltre le proprie condizioni riguardo la terminazione. La classe Stage implementa una logica di terminazione di base, lasciando comunque aperta la possibilitá per gli stadi risolutivi concreti di estenderla. In questa logica di base, si indica uno stadio come terminato quando una delle seguenti condizioni si verifica:

(1) Il numero di step eseguiti raggiunge un valore di soglia;
(2) Il numero di step eseguiti *senza miglioramenti della soluzione ottima* raggiunge un valore soglia.

2. Stato (StageState)

Il concetto di stato interno viene rappresentato nel framework da una gerarchia di classi che ha come radice la classe StageState.

Questa classe raccoglie a fattor comune quelle variabili di stato che ogni di cui ogni stadio risolutivo necessita sicuramente, tra cui:

- un contatore che mantiene il numero di step eseguiti;
- la soluzione correntemente in esame;
- la miglior soluzione finora determinata, e lo step a cui questa é stata determinata;
- altre informazioni statistiche utili per l'analisi del risolutore.

In questa classe é implementata inoltre la logica di base relativa all'aggiornamento di queste variabili.

Lo step risolutivo operato da uno stadio consiste principalmente nella modifica della soluzione corrente contenuta nello stato. Avvenuto ció, lo stadio informa lo stato interno dell'avvenuta modifica, e questo scatena la logica di aggiornamento di tutte le altre variabili di stato.

L'operazione fondamentale di aggiornamento dello stato al termine di uno step risolutivo riguarda l'eventuale raggiungimento di una nuova soluzione migliorante. Questa parte di logica puó essere descritta nella maniera seguente, tralasciando l'aggiornamento delle variabili statistiche secondarie:

Algorithm 2 StageState: aggiornamento post-step.

```
1:  function STEPDONE
2:      currentStep ← currentStep + 1
3:      if ISFEASIBLE(currentSolution) then
4:          curP = PROFIT(currentSolution)                ▷ Primary objective
5:          bestP = PROFIT(bestSolution)
6:          curTS = TOTALTIMESPAN(currentSolution)        ▷ Secondary objective
7:          bestTS = TOTALTIMESPAN(bestSolution)
8:          improved = (curP > bestP)                     ▷ Profit
9:                ∨((curP ≡ bestP) ∧ (curTS < bestTS))    ▷ Time span
10:         if improved then
11:             bestSolution ← currentSolution
12:             nBest ← nBest + 1
13:         end if
14:     end if
15: end function
```

É naturale immaginare che tipi di stadi risolutivi diversi abbiano necessitá di mantenere altre variabili di stato oltre a quelle di base; il framework consente di rispondere a questa esigenza, attraverso l'estensione arbitraria della classe StageState.

Questo avviene, ad esempio, per implementare stadi risolutivi basati su Tabu Search: per questi ultimi é necessario, infatti, aggiungere allo stato di base variabili di stato aggiuntive (ad esempio, quelle relative alla tabu-list), nonché la logica necessaria all'aggiornamento di queste.

3. Generatore di soluzione iniziale (InitialSolutionGenerator)

Introdotto il concetto di stadio risolutivo e del proprio stato interno, resta da definire all'interno del framework un modo per generare una soluzione di partenza per il risolutore.

Questo compito é delegato ad un componente che risponde all'interfaccia InitialSolutionGenerator.

Il risolutore viene infatti configurato principalmente con un generatore di soluzione iniziale ed uno stadio principale detto di *root-level*.

Nella fase di inizializzazione, il motore risolutivo compie i seguenti passi:

(1) Viene creata una soluzione vuota;
(2) Viene interrogato il generatore di soluzione iniziale che, modificando la soluzione vuota, la porta, secondo la particolare politica di inizializzazione che implementa, ad uno stato iniziale;
(3) Viene interrogato lo stadio risolutivo di root-level, eseguendo iterativamente gli step risolutivi.

3.1. Packed Groups. Il generatore iniziale impiegato dal nostro algoritmo é denominato *Packed Groups*.

Esso si fonda sull'osservazione che il BTP consiste nella sovrapposizione di un problema di carico e un problema di routing con time-window. Si e' osservato che il problema di carico (ovvero assegnare i gruppi di turisti ai Bus) é tendenzialmente piú semplice del problema di routing dei veicoli; questo perché il problema riguardo il carico é caratterizzato da minori gradi di libertá e da vincoli piú semplici.

Pertanto, si é pensato di inizializzare la soluzione assegnando in un primo passo tutti i turisti ai Bus, minimizzando il numero di Bus richiesti.

3.1.1. *Assegnazione dei Gruppi con bin-packing approssimato.* Per far ció, é stato impiegata una strategia approssimata per il bin-packing denominata *first-fit-decreasing* (FFD)[1]:

Si ordinano i gruppi di turisti per dimensione decrescente, e si assegna ciascun gruppo ad un Bus che lo puó contenere, prediligendo il Bus di indice minimo. Se non esiste un Bus inizializzato con sufficiente spazio disponibile, ne viene allocato uno nuovo.

Per ragioni di semplicitá, questo inizializzatore presuppone che tutti i Bus siano omogenei per capacitá, ovvero che la formulazione del problema contempli un solo BusGroup.

Sebbene si tratti di un algoritmo approssimato, il FFD é caratterizzato dal fatto che, nel caso peggiore, generi un'assegnazione che richiede $11/9 \times OPT$ bin, dove con OPT si indica il numero ottimale di bin (ottenibile quindi con algoritmi esatti, ben piú complessi).

L'ottimo worst-case-bound é accompagnato da sorprendenti risultati anche in termini statistici: in problemi con 90 elementi uniformemente distribuiti nell'intervallo $(0, 10^6)$, con bin di capacitá 10^6, l'FFD impiega una media di 47.733 bin, mentre un algoritmo esatto ne impiega in media 47.860. Inoltre, le soluzioni trovate da FFD sono ottimali per il 94.694% dei casi.[2]

Resta comunque la (remota, per problemi non eccessivamente saturi) possibilitá che il passo di FFD richieda un numero di Bus maggiori di quelli disponibili. In questo caso, l'inizializzatore fallisce ed il risolutore non puó essere impiegato.

É stato verificato che questo fallimento non avvenga nell'insieme di problemi di test.

Si giunge a questo punto disponendo di un certo numero di Bus, corrispondenti ad altrettante Route, e dell'assegnazione di tutti i gruppi di turisti a disposizione. Tuttavia, queste Route al momento non visitano alcun POI.

3.1.2. *Assegnazione 1-POI iniziale (asta).* In questo secondo passo il generatore di soluzione iniziale provvede a soddisfare la richiesta dei turisti in maniera preliminare, assegnando un POI a ciascuna Route. Per far ció viene impiegato un meccanismo *ad asta*, nel quale si considera una pool $\mathbb{R}$ di Route contenente inizialmente l'intero insieme di Route, ed una pool $\mathbb{P}$ di POI disponibili contenente inizialmente tutti i POI visitabili.

Ad ogni passo, si eseguono le seguenti operazioni, finché la pool di Route non é vuota:

(1) Si determina la Route $r \in \mathbb{R}$ ed il POI $p \in \mathbb{P}$ per i quali l'assegnazione di p ad r determina il massimo incremento di profitto;
(2) Il POI p viene assegnato alla Route r;
(3) La Route r viene rimossa dalla pool $\mathbb{R}$;

---

[1] Silvano Martello e Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations (Wiley Series in Discrete Mathematics and Optimization)*. 1990.

[2] Richard E. Korf. «A New Algorithm for Optimal Bin Packing». In: *AAAI Proceedings*. 2002.

(4) Se il POI p ha raggiunto con quest'ultima assegnazione il massimo numero di visite possibili (secondo il vincolo specificato nel problema), esso viene rimosso dalla pool $\mathbb{P}$ di POI visitabili.

3.1.3. *Considerazioni.* L'assegnazione dei gruppi di turisti alle Route é fatta ignorando completamente le preferenze espresse riguardo i POI: é perció possibile che una Route contenga inizialmente gruppi di turisti con preferenze incompatibili.

Si ritiene che questa evenienza rappresenti un fattore negativo sostanzialmente trascurabile rispetto al grande vantaggio di avere un'assegnazione dei gruppi compatta e che, salvo casi patologici, riesce a rispettare il vincolo sul numero massimo di Bus impiegabile.

Nello stesso senso, é vero che l'assegnazione iniziale di un solo POI a ciascuna Route puó sembrare limitativa, ma anche in questo caso il vantaggio é di aver garanzie sulla soluzione iniziale:

- Supponendo che la formulazione del problema non contenga definizioni di POI *patologici*, ovvero POI non raggiungibili dal Depot senza violazioni delle time-window, si ha la garanzia che la soluzione iniziale non puó violare appunto le finestre temporali;
- Il vincolo sul massimo numero di visite é sicuramente rispettato;

Supponendo quindi che la fase di bin-packing abbia successo, si ottiene *una soluzione iniziale semplice, compatta ed ammissibile*. Questo é strettamente necessario: il framework, infatti, é fondato sull'assunzione (che viene verificata) che qualsiasi stadio risolutivo debba partire da una soluzione ammissibile e dare origine ad una soluzione finale ammissibile.

4. Stadi disponibili

Generata la soluzione iniziale, il compito della risoluzione ricade unicamente sullo stage di root-level con cui il risolutore e' configurato.

Vengono qui presentati i due tipi di stadio implementati, e come sia possibile in particolare *definire un risolutore multistadio basato su Tabu Search* (configurazione utilizzata nel nostro risolutore sperimentale).

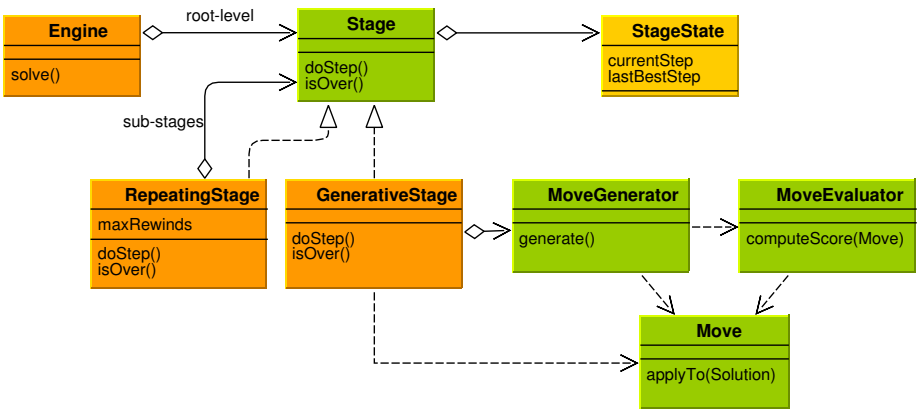


FIGURA 2. Organizzazione concettuale degli stadi risolutivi.

4.1. Stadio ripetitore (RepeatingStage). Questo tipo di stadio permette di definire risolutori multi-stadio, ovvero risolutori costituiti da una sequenza di stadi risolutivi in generale eterogenei.

Lo stadio viene configurato con una sequenza di stadi che esso pilota, denominati *sub-stage*; ad ogni passo risolutivo, il compito di eseguire lo step viene delegato al sub-stage correntemente attivo.

Quando si verifica la condizione di terminazione per un sub-stage, lo stadio ripetitore seleziona il sub-stage successivo, inizializzandolo e trasferendo ad esso la soluzione ottima corrente.

Lo stadio si chiama ripetitore perché é possibile configurarlo in generale per "riavvolgere" la sequenza di stadi al termine dell'ultimo: in questo caso, terminato l'ultimo stadio risolutivo, la parola passa nuovamente al primo. Il massimo numero di riavvolgimenti é un parametro di configurazione: specificando 0 come valore si ottiene la semplice esecuzione in sequenza degli stadi, senza ripetizioni.

4.2. Stadio generativo (GenerativeStage). Questo tipo di stadio permette di realizzare risolutori che esplorano lo spazio del problema modificando la soluzione corrente per mezzo di mosse. Questo tipo di esplorazione viene chiamata local-neighbourhood search, in quanto le perturbazioni della soluzione corrente che le mosse generano determinano per l'appunto un vicinato, ovvero una piccola regione dello spazio del problema che si sviluppa in prossimitá della soluzione corrente.

Lo stadio generativo delega la generazione vera e propria delle mosse ad un componente denominato Move Generator, che viene specificato in fase di configurazione; quest'ultimo, a sua volta, delega la politica di selezione della mossa vincente tra tutte quelle generate ad un componente denominato MoveEvaluator.

Questa chiara suddivisione dei compiti tra componenti diversi conferisce al framework una certa modularitá.

Per dettagli riguardanti le varianti di generatori di mosse implementate si rimanda al capitolo 7. Questo capitolo compare dopo l'introduzione al nostro algoritmo, in quanto molte delle scelte progettuali riguardanti i generatori di mosse vengono illustrate utilizzando quest'ultimo come banco di prova.

5. Supporto del framework al Tabu Search

Il nostro interesse principale é la definizione di un risolutore fondato sul Tabu Search (rif. sez. 1).

Nell'ottica del framework questo puó essere realizzato come:

- un GenerativeStage,
- che utilizza come stato interno un TabuState,
- e che utilizza per il ranking delle mosse un'implementazione di MoveEvaluator denominata TabuMoveEvaluator.

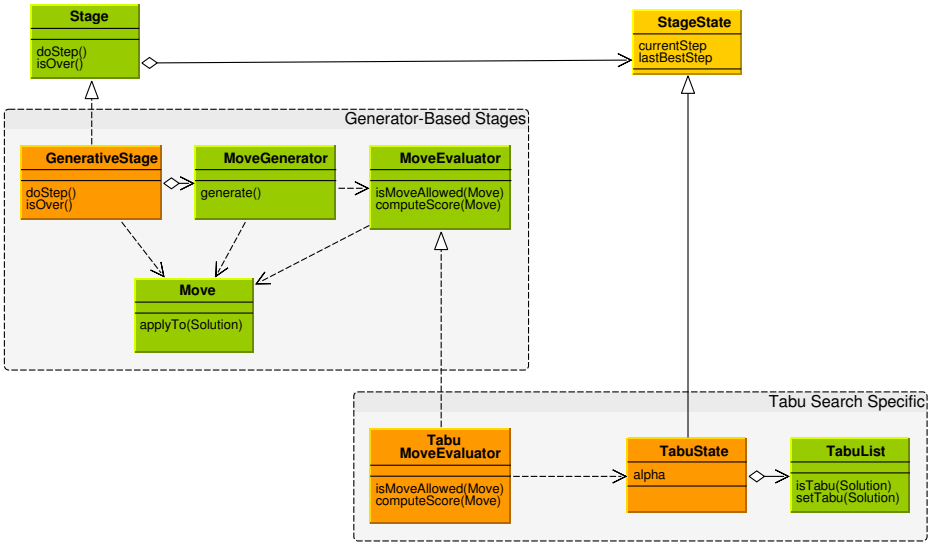


FIGURA 3. Supporto del framework a stadi Tabu Search.

5.1. TabuState. La classe TabuState rappresenta una specializzazione dello stato interno di uno stadio risolutivo con l'aggiunta delle variabili di stato e della logica di aggiornamento necessari ad implementare la politica del Tabu Search.

5.1.1. *Tabu List.* Esso contiene innanzitutto una tabu-list, che nel nostro caso viene implementata come una lista di soluzioni da considerare momentaneamente inaccessibili. La durata dello stato di tabu per una soluzione (chiamato in letteratura *tenure*) é espresso in termini di numero di step risolutivi e specificato dal parametro configurabile θ.

La tabu-list *rappresenta il concetto di memoria a breve termine* tipica del Tabu Search: essa serve principalmente ad oltrepassare gli estremanti locali della funzione obiettivo ed evitare cicli nell'esplorazione delle soluzioni.

5.1.2. *Modulazione* α. Per mettere in atto la politica di intensificazione e diversificazione, concetto molto importante nell'ambito del Tabu Search, si utilizza una strategia molto semplice ma efficace che consiste nel modulare in maniera dinamica la tendenza del risolutore ad uscire dalla regione ammissibile.

Questo attraversamento controllato di regioni non ammissibili permette di trovare "scorciatoie" nello spazio del problema e di esplorarne regioni altrimenti inaccessibili.

Nella classe TabuState il supporto a questo avviene mediante una variabile di stato, detta α, che rappresenta un coefficiente di penalizzazione per le soluzioni non ammissibili. Questa variabile viene modificata sulla base di una specifica regola di aggiornamento, controllata da un parametro configurabile detto β e da un'altra variabile di stato, detta $streak$.

Lo *streak* é il numero dei passati step consecutivi che non hanno comportato cambiamenti nell'ammissibilitá della soluzione: se, ad esempio, negli ultimi 10 step la soluzione é rimasta ammissibile, *streak* assume valore 10.

La modifica di α avviene quando $streak >= \beta$. In tal caso si é determinata sperimentalmente la seguente politica di aggiornamento:

- Se la sequenza é composta da soluzioni ammissibili, si applica la formula che riduce α, in modo da iniziare a favorire l'esplorazione di soluzioni non ammissibili: $\alpha \leftarrow max(0.01, (\alpha \cdot 0.7))$;
- Se la sequenza é composta da soluzioni non ammissibili, si applica la formula che aumenta α, in modo da favorire il processo opposto: $\alpha \leftarrow \alpha \cdot 1.2$

Si nota come la rapiditá con cui si esce dalla regione ammissibile sia maggiore di quella con cui si rientra.

Questo perché si é osservato che spesso, giunti in regione non ammissibile, é necessaria una certa permanenza in tale regione prima di trovare un'area migliore per il rientro: questo significa rallentare il rientro.

Il processo opposto, quello di uscita, avviene invece a causa del fatto che l'area di esplorazione non é fertile: in questo caso é opportuno uscire velocemente. Tuttavia, é in questo caso importante limitare l'appetibilitá massima delle soluzioni non ammissibili, per evitare di generare soluzioni irrimediabilmente compromesse: il limite é stato fissato al valore 0.01, corrispondente quindi ad un fattore di penalizzazione di un centesimo.

Come nota finale, si informa che le formule di aggiornamento di α possono essere modificate; tuttavia, al momento, esse sono determinate a tempo di compilazione, e quindi sono comuni a tutti i TabuState impiegati.

5.2. TabuMoveEvaluator. Per implementare la selezione di mosse tipica del tabu-search, é stato implementato un particolare MoveEvaluator che fa uso del TabuState per effettuare il ranking delle mosse provenienti dai generatori.

Innanzitutto, questo MoveEvaluator scarta le mosse provenienti dai generatori se esse comportano lo spostamento in una soluzione che é al momento marcata come tabu nella tabu-list.

Per effettuare il ranking delle mosse accettate, viene innanzitutto determinata, a partire dalla soluzione di partenza e dalla mossa in esame, l'ipotetica soluzione di destinazione a cui la mossa darebbe origine.

Indicando con $destSolution$ questa ipotetica soluzione, il TabuMoveEvaluator calcola il punteggio per la mossa in esame con la seguente formula:

$$score(move) = profit(destSolution) - \alpha \cdot infeasibilityIndex(destSolution)$$

, dove

$profit()$**:** indica il calcolo del profitto della soluzione

α**:** indica il fattore di modulazione corrente (variabile letta dal TabuState)

$infeasibilityIndex()$**:** indica il calcolo dell'indice di non-ammissibilitá della soluzione. Per la descrizione di come avviene il calcolo di quest'ultimo si rimanda alla sezione successiva.

Il punteggio di una mossa avviene quindi considerando il profitto che si otterrebbe applicando la mossa, e penalizzando tale profitto in base all'entitá dell'eventuale violazione dei vincoli. Dato che la penalizzazione dipende anche dal parametro α, é possibile, nonché probabile, che in corrispondenza di step diversi lo stesso indice di non-ammissibilitá si traduca in penalizzazioni di entitá diversa.

5.2.1. *Indice di non ammissibilitá (Infeasibility Index).* Il calcolo dell'indice di non ammissibilitá di una soluzione serve sostanzialmente a stimare in che misura questa stia violando i vincoli del problema.

Si ritiene che una buona definizione di questo indice sia essenziale per ottenere un'esplorazione efficiente; un cattivo indice di non-ammissibilitá introduce inevitabilmente degli squilibri (*bias*) nel ranking delle mosse.

Ai fini di assicurare una certa imparzialitá, si é deciso di progettare questo indice in modo che esso sia legato in maniera pressoché proporzionale al profitto della Route: il calcolo qui presentato, infatti, cerca di stimare la parte di profitto raccolta "illecitamente" grazie alla violazione dei vincoli.

Algorithm 3 Infeasibility Index

```
1:  function INFEASIBILITYINDEX(solution)
2:      return ∑_route ROUTEINFINDEX(route)
3:  end function
```

Dove l'infeasibility index di una route viene calcolato come:

```
4:  function ROUTEINFINDEX(route)
5:      infIndex ← 0
6:      avgProft = PROFIT(route)/nPOIs
7:      loadRatio = LOAD(route)/CAPACITY(bus)
8:      for all poi ∈ POIS(route) do
9:          if arrival time to poi is violated then                ▷ break TW?
10:             penalty = (arrival − et)/(lt − et)
11:         else if load is violated then                          ▷ overload?
12:             penalty = overload
13:         end if
14:         poiProfit ← profit for poi
15:         if poiProfit ≡ 0 then
16:             poiProfit ← avgProfit
17:         end if
18:         infIndex ← infIndex + poiProfit · penalty
19:     end for
20:     if arrival to depot is violated then                       ▷ break depot TW?
21:         infIndex ← infIndex + PROFIT(route) ∗ (arrival/lt_depot)
22:     end if
23: end function
```

CAPITOLO 6

Struttura dell'algoritmo

Dato che il BTP coinvolge numerosi vincoli ed entitá, é stato da subito tentato di trovare una direzione opportuna secondo la quale partizionare il problema in sotto-problemi piú facili da risolvere.

La direzione secondo la quale si é deciso di suddividere il problema é stata individuata considerando la risoluzione del BTP come la risoluzione contestuale di due problemi: uno di carico, che si occupa di assegnare i gruppi di turisti ai Bus, ed uno di routing, che si occupa degli aspetti topologici dei percorsi intrapresi (Route).

In questa visione, l'osservazione principale che ha gettato le basi dell' algoritmo riguarda la sostanziale complementaritá dei due sotto-problemi. La prima faccia di questa relazione consiste nel fatto che la determinazione del percorso intrapreso da un Bus ha senso solo se per tale si dispone di un'assegnazione pre-esistente di gruppi di turisti *sufficientemente* consistente. La considerazione analoga é che la qualitá dell'assegnazione di turisti ai Bus é valutabile solo nell'ottica del percorso intrapreso.

Strategia

Questo ha portato sostanzialmente alla decisione di operare una serie di *risoluzioni parziali*, in cui alternativamente si cerca di ottenere progressi nei due problemi coinvolti. Questa commutazione periodica permette di mantenere un equilibrio tra il grado di definitivitá della soluzione nei riguardi delle due problematiche.

L'algoritmo di risoluzione da noi presentato segue quindi un approccio multi-stadio che coinvolge 3 stadi risolutivi basati su Tabu Search.

Il primo stadio, denominato *Groups*, si occupa di ottimizzare la distribuzione dei gruppi nei Bus, attraverso mosse di spostamento o trasferimento dei turisti insoddisfatti.

Il secondo stadio, denominato *Routes*, si occupa del miglioramento dei percorsi intrapresi, ed opera attraverso mosse di inserimento di POI non ancora visitati, scambio di POI tra Route differenti, e di suddivisione ed accorpamento di Route.

Il terzo stadio, denominato *Opt1*, si occupa della riduzione della durata temporale delle Route, attraverso mosse di scambio intra-route di POI (comunemente chiamate 1-OPT). Per questo compito di post-ottimizzazione intermedia é stato dedicato uno stadio separato a causa dell'impiego intensivo di risorse. In questo modo é possibile mantenere questo impiego di risorse sotto controllo attraverso regolazioni sui parametri che controllano le condizioni di terminazione dello stadio.

Viene impiegato un'ulteriore stadio, di tipo Repeating, detto di root-level, che si occupa del passaggio allo stadio successivo quando le condizioni di terminazione per lo stadio corrente vengono soddisfatte. É opportuno sottolineare che lo stadio di tipo Repeating é configurabile appunto con un numero massimo di ripetizioni (rewind) consentiti, e quindi in generale esso, terminata l'esecuzione del terzo stadio, procede alla riattivazione del primo stadio.

In questo modo si mette in pratica la strategia sopra indicata, secondo la quale i progressi riguardo il sotto-problema di carico vengono propagati dal primo al secondo stadio, che si dedica al sotto-problema di routing, e cosí via.

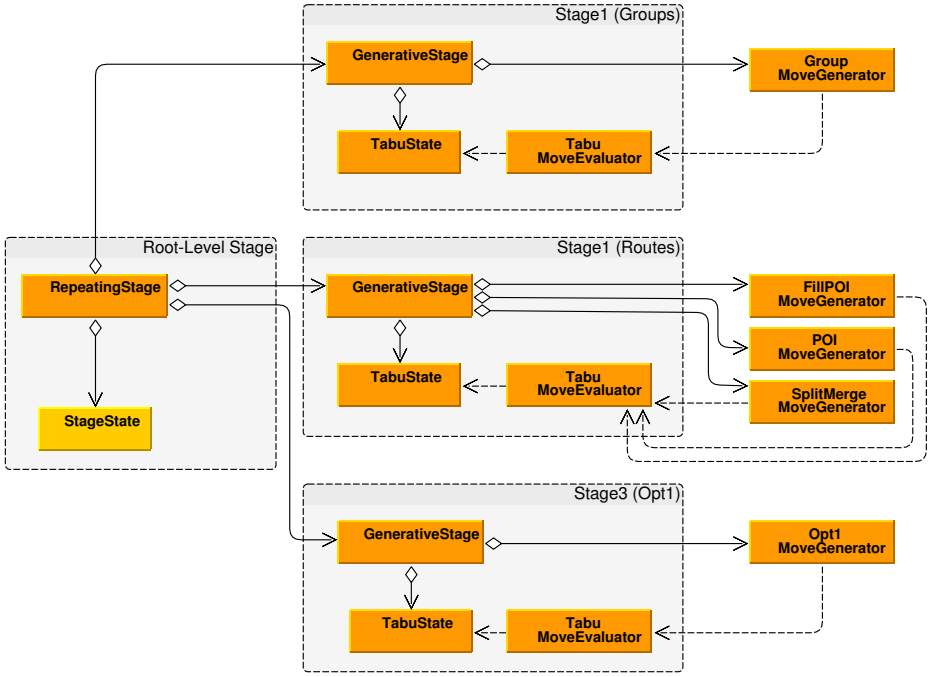


FIGURA 1. Struttura multi-stadio dell'algoritmo.

Mentre la struttura del risolutore e l'impiego del Tabu Search determinano la strategia risolutiva di alto livello, i generatori di mosse caratterizzano invece la tecnica di esplorazione a basso livello. Nel prossimo capitolo vengono descritti in dettaglio tutti i generatori di mosse impiegati.

CAPITOLO 7

Generatori di mosse

1. Introduzione

I generatori di mosse sono un componente essenziale per quella famiglia di risolutori euristici che si fonda sulla ricerca locale.

Questo tipo di risoluzione opera per passi, analizzando ad ogni iterazione un sottoinsieme dello spazio delle soluzioni, che é denominato vicinato perché esso é ottenuto perturbando la soluzione corrente mediante un insieme di mosse.

La generazione di mosse rappresenta quindi uno dei fattori che caratterizzano maggiormente l'euristica impiegata, mentre la strategia che guida l'esplorazione vera e propria va a caratterizzare la meta-euristica.

Il compito di generare, data una soluzione corrente, un insieme di mosse che la perturbano, viene rappresentato nel framework da una gerarchia di classi che implementano l'interfaccia denominata, appunto, MoveGenerator. Tutte le mosse generate appartengono alla gerarchia di classi che ha come radice l'interfaccia denominata Move.

2. Panoramica dei generatori

Rimandando alle sezioni specifiche per i dettagli sui vari generatori, si riportano qui le informazioni riassuntive riguardo essi.

TABELLA 1. Panoramica delle mosse fornite dai diversi generatori.

Generatore	Mosse generate									
	Swap	Transfer	Insert POI		Swap POI			Split	Merge	Opt1
	Group	Group	Normal	Replace	1-1	0-1	1-0	Route	Route	
Group M.G.	✓	✓								
FillPOI M.G.			✓	✓						
POIMove M.G.					✓	✓	✓			
SplitMerge M.G.								✓	✓	
Opt1 M.G.										✓

3. Group Move Generator

3.1. Scopo. Lo scopo di questo generatore é di perturbare la distribuzione di gruppi di turisti tra le varie Route esistenti. Si ottiene quindi la possibilitá di ridistribuire sia il carico che i profitti delle Route.

Viene inizialmente selezionata una pool di Route che rispettano determinate caratteristiche. Per ciascuna Route, viene determinata una pool di Group "insoddisfatti" dalla selezione dei POI da visitare.

Per ogni coppia di Group insoddisfatti provenienti da Route diverse, viene generata una mossa di scambio gruppo.

Inoltre, per ciascun Group insoddisfatto, viene generata anche una mossa di trasferimento alle altre Route della pool. Questo trasferimento potrebbe lasciare la Route di origine completamente scarica: in questo caso, la mossa di trasferimento provvede automaticamente alla cancellazione della Route.

3.2. Mosse generate. Le mosse generate sono:

- Swap Group (RouteA, GroupA, RouteB, GroupB):
  Scambia i gruppi di turisti indicati tra le due Route indicate (distinte).
- Transfer Group (RouteA, GroupA, RouteB):
  Trasferisce il Group indicato dalla Route A alla Route B.

3.3. Politica di generazione delle mosse. L'algoritmo di generazione é il seguente:

Algorithm 4 Group Move Generator

Require: MAX_GROUPS ▷ max n. of unsatisfied Groups per Route
Require: MAX_LP ▷ max n. of low-profit Routes
Require: MAX_HL ▷ max n. of high-load Routes

function GENERATEGROUPMOVES
  $\mathbb{M} \leftarrow \emptyset$
  $lpRoutes$ = SELECTLOWPROFITROUTES(MAX_LP)
  $hlRoutes$ = SELECTHIGHLOADROUTES(MAX_HL)
  for all DISTINCT($routeA, routeB$) $\in$ ($lpRoutes \cup hlRoutes$) **do**
    $groupsA$ = SELECTUNSATISFIEDGROUPS($routeA$, MAX_GROUPS)
    $groupsB$ = SELECTUNSATISFIEDGROUPS($routeB$, MAX_GROUPS)
    for all $gA \in groupsA$ **do**
      $\mathbb{M} \leftarrow \mathbb{M} \cup$ TRANSFERGROUP($routeA, gA, routeB$)
    end for
    for all $gB \in groupsB$ **do**
      $\mathbb{M} \leftarrow \mathbb{M} \cup$ TRANSFERGROUP($routeB, gB, routeA$)
    end for
    for all $(gA, gB) \in (groupsA, groupsB)$ **do**
      $\mathbb{M} \leftarrow \mathbb{M} \cup$ SWAPGROUP($routeA, gA, routeB, gB$)
    end for
  end for
  return $\mathbb{M}$
end function

Dove le specifiche funzioni sono:

function SELECTLOWPROFITROUTES(maxN)
  $routes$ = SORTBY($\mathbb{R}, r \mapsto profit(r), ASC$)
  return TRIMTOSIZE($routes, maxN$)
end function
function SELECTHIGHLOADROUTES(maxN)
  $routes$ = SORTBY($\mathbb{R}, r \mapsto load(r), DESC$)
  return TRIMTOSIZE($routes, maxN$)
end function
function SELECTUNSATISFIEDGROUPS(route, maxN)
  $groups \leftarrow$ GETGROUPS($route$)
  $groups \leftarrow$ SORTBY($groups, g \mapsto profitInRoute(g, route), ASC$)
  return TRIMTOSIZE($groups, maxN$)
end function

3.4. Parametri.

MAX_GROUPS: Dimensione massima della pool di Group insoddisfatti per ciascuna Route
MAX_LP: Dimensione massima della pool di Route low-profit
MAX_HL: Dimensione massima della pool di Route high-load

3.5. Effetto tipico. L'effetto tipico del generatore é il seguente:

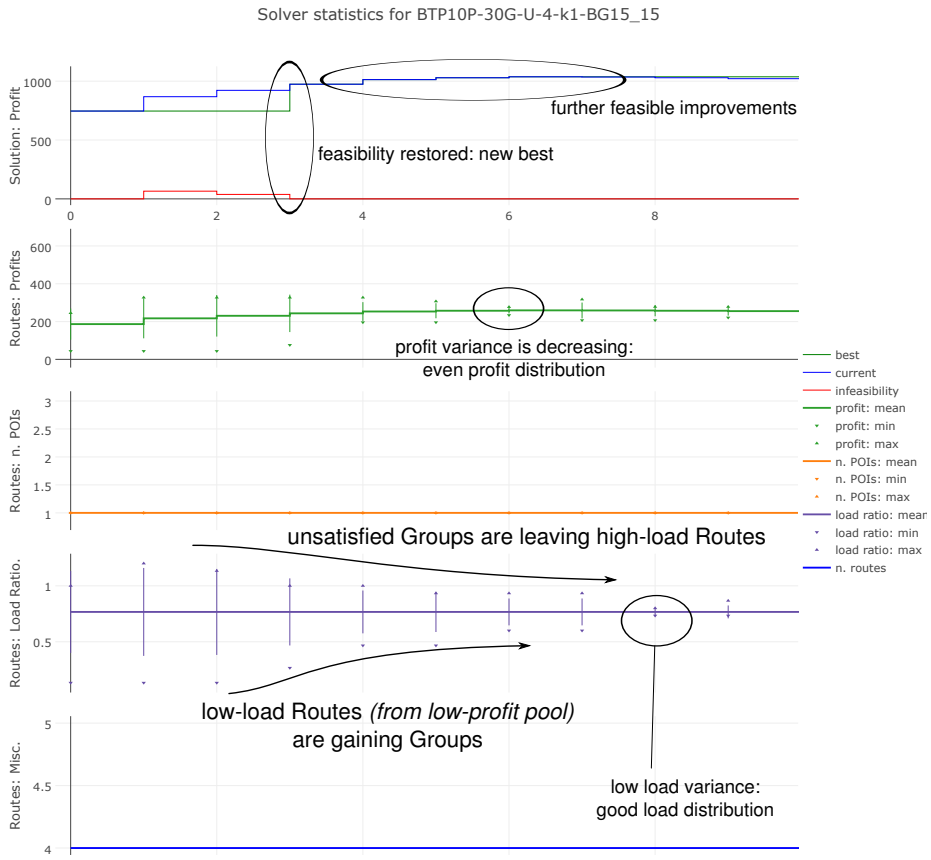


FIGURA 1. Sequenza di mosse provenienti da un Group Move Generator accettate dal risolutore nei passi iniziali. Si nota come, dopo un'iniziale redistribuzione dei carichi (alta varianza di load ratio), l'effetto delle mosse fa si che i gruppi insoddisfatti lascino progressivamente le Route sovraccariche per trasferirsi nelle Route scariche. La disponibilitá di Route scariche é fornita euristicamente dalla low-profit pool: é infatti probabile che il basso profitto di una Route sia attribuibile ad un basso carico di turisti. Questo in virtú del presupposto su cui si fonda il risolutore: evitare in ogni stadio condizioni di elevata inconsistenza tra i POI visitati da una Route ed i gruppi all'interno di essa.

3.6. Note. Nelle sue versioni iniziali, il generatore era molto piú rudimentale e soffriva di alcuni problemi.

Innanzitutto, venivano generate solo mosse di scambio tra gruppi, e non mosse di trasferimento. In secondo luogo, i gruppi considerati negli scambi provenivano tutti da un unica pool di Route low-profit.

L'intento iniziale era di scambiare gruppi insoddisfatti tra Route a basso profitto, mantenendo comunque una certa uniformitá nella distribuzione dei carichi. La considerazione di partenza si fonda, infatti, sul fatto che le Route siano in contesa per i POI disponibili

(ricordando che esiste un vincolo sul massimo numero di visite per un POI), e che questa contesa sia una rappresentazione delle preferenze dei singoli gruppi che esse contengono. Una distribuzione eccessivamente squilibrata dei carichi tende a tradursi in un'organizzazione gerarchica delle Route, che vede quelle piú cariche influenzare il risolutore nella fase di assegnazione dei POI da visitare, a discapito delle altre meno cariche, per le quali l'assegnazione di POI viene vista dal risolutore come un processo marginale. Inizialmente, quindi, si pensava che il divieto di trasferire gruppi fosse una garanzia sull'equilibrio del risolutore.

Tuttavia, osservando in dettaglio il processo di risoluzione, é stato identificato ed analizzato un problema sulla dinamica di esplorazione del vicinato: gli scambi generati non permettevano alle Route di uscire dalla pool low-profit, con la conseguenza che un'area molto vasta del vicinato veniva ignorata dal risolutore.

Si é pensato di introdurre quindi una seconda pool "complementare", che, in virtú della maniera in cui era generata donasse dinamicitá al processo di selezione. Per far ció é stata effettuata la seguente osservazione: gli scambi tra i gruppi servono a migliorare il profitto (e quindi agiscono sulla route low-profit), mentre l'effetto collaterale é la modifica dei carichi. Questo ha fatto intuire che la nuova pool complementare dovesse essere determinata sulla base dei carichi.

Il punto di svolta é stato rappresentato dall'osservazione che lo scambio tra gruppi non modifica sostanzialmente i carichi delle Route coinvolte: é stata introdotta, pertanto, in maniera speculativa, la mossa di trasferimento.

La seconda pool "complementare" é stata determinata dalla considerazione che la pool low-profit é tendenzialmente scarica, in virtú della filosofia su cui si basa il risolutore: limitare l'eccessivo squilibrio tra i gruppi assegnati ad una Route ed i POI che essa visita. In questo senso, se i POI assegnati sono sufficientemente coerenti con i gruppi che li visitano (in termini di profitto), é naturale pensare che una Route tendenzialmente raccoglie poco profitto a causa del basso carico.

Questo ha fatto decidere di utilizzare la politica high-load per la seconda pool di Route.

Questa decisione ha migliorato il processo risolutivo: analizzando i casi in cui lo stadio che fa uso del Group Move Generator trova una soluzione fattibile migliorante, si osserva che in gran parte questo succede dopo aver applicato almeno una mossa di trasferimento di un gruppo da una Route appartenente alla high-load pool ad una Route appartenente alla low-profit pool.

4. Fill POI Generator

4.1. Scopo. Lo scopo principale di questo generatore é l'introduzione nelle Route di POI non ancora visitati.

Per far ció, vengono selezionati un certo numero di POI non ancora visitati, e, per ciascuno di essi, un certo numero di Route che "competono" per aggiudicarselo.

Si ottiene quindi un insieme di coppie del tipo (POI, Route): per ciascuna di queste coppie vengono generati tutti i possibili inserimenti (con e senza rimpiazzo) del POI ad ogni posizione della Route.

4.2. Mosse generate. Le mosse generate sono:

- Insert POI Without Replacement (Route, POI, Position):
  Il POI viene inserito nella Route alla posizione specificata.
- Insert POI With Replacement (Route, POI, Position):
  Il POI che attualmente é nella Route alla posizione specificata viene sostituito con il POI indicato.

4.3. Politica di generazione delle mosse. Viene inizialmente selezionata una pool contentente al massimo un numero configurabile di POI non ancora visitati, prediligendo quelli con distanza inferiore dal Depot.

Viene poi selezionata una pool di Route, di dimensione massima configurabile, denominata low-timespan, prediligendo le Route con timespan inferiore. Questa pool di route é quindi indipendente dai POI selezionati.

Successivamente, per ogni POI selezionato, viene determinata una pool di Route , di dimensione massima configurabile, denominata highly-profitable, prediligendo le Route che, ricevendo il POI selezionato, ricaverebbero il massimo beneficio in termini di profitto.

Per ogni POI selezionato, quindi, viene creata una pool data dall'unione della pool statica (low-timespan) con quella dinamica (highly-profitable). Per ciascuna Route in questa pool, viene generata una mossa di inserimento del POI (con e senza rimpiazzo) ad ogni posizione della Route.

L'algoritmo di generazione é il seguente:

Algorithm 5 Fill POI Generator

Require: MAX_POIS ▷ max n. of nearest POIs to the Depot
Require: MAX_LT ▷ max n. of low-timespan Routes
Require: MAX_HP ▷ max n. of highly-profitable Routes per POI

function GENERATEFILLPOIMOVES
  $\mathbb{M} \leftarrow \emptyset$
  $pois =$ SELECTNEARESTPOIS(MAX_POIS)
  $ltRoutes =$ SELECTLOWTIMESPANROUTES(MAX_LT)
  for all $poi \in pois$ **do**
    $hpRoutes =$ SELECTHIGHLYPROFITABLEROUTES(poi, MAX_HP)
    for all $route \in (ltRoutes \cup hpRoutes)$ **do**
      for all $pos \in [0, size(route) - 1]$ **do**
        $\mathbb{M} \leftarrow \mathbb{M} \cup$ INSERTPOI($route$, poi, pos)
        $\mathbb{M} \leftarrow \mathbb{M} \cup$ INSERTPOIWITHREPLACEMENT($route$, poi, pos)
      end for
    end for
  end for
  return $\mathbb{M}$
end function

Dove le specifiche funzioni sono:

function SELECTNEARESTPOIS(maxN)
  $pois =$ SORTBY($\mathbb{P}$, $p \mapsto distance(p, depot)$, ASC)
  return TRIMTOSIZE($pois$, $maxN$)
end function
function SELECTLOWTIMESPANROUTES(maxN)
  $routes =$ SORTBY($\mathbb{R}$, $r \mapsto timespan(r)$, ASC)
  return TRIMTOSIZE($routes$, $maxN$)
end function
function SELECTHIGHLYPROFITABLEROUTES(poi, maxN)
  $routes =$ SORTBY($\mathbb{R}$, $r \mapsto profit(r, poi)$, $DESC$)
  return TRIMTOSIZE($routes$, $maxN$)
end function

4.4. Parametri.

MAX_POIS: Dimensione massima della pool di POI vicini al depot
MAX_LT: Dimensione massima della pool di Route low-timespan
MAX_HP: Dimensione massima della pool di Route highly-profitable per ciascun POI selezionato

4.5. Effetto tipico. L'effetto tipico del generatore é il seguente:

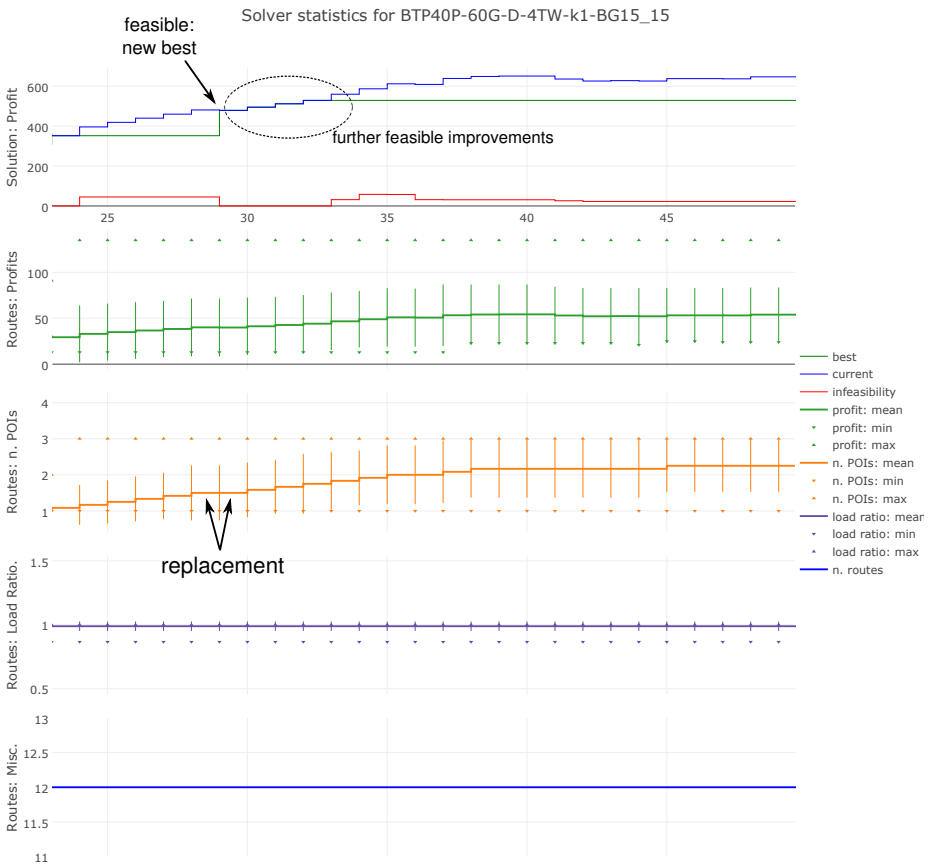


FIGURA 2. Sequenza di mosse provenienti da un Fill POI Generator accettate dal risolutore nei passi iniziali. Si nota l'aumento progressivo del numero di POI nelle Route e del profitto. In corrispondenza del passo indicato, si nota come l'inserimento con rimpiazzo abbia ripristinato l'ammissibilitá della soluzione.

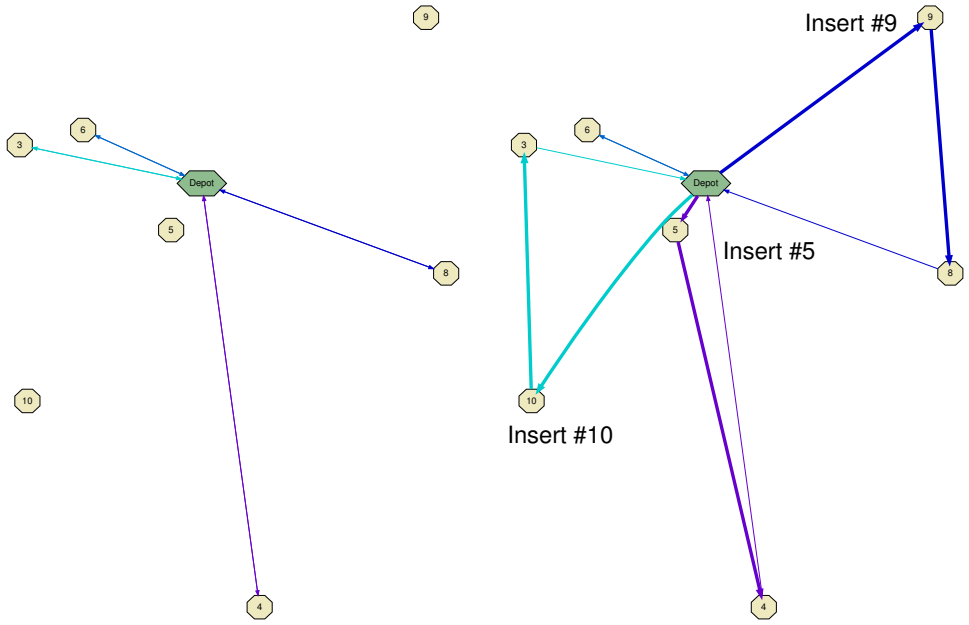


FIGURA 3. Esempio dell'applicazione di 3 mosse consecutive di inserimento senza rimpiazzo, applicate nei passi iniziali di risoluzione. Si nota infatti come siano ancora presenti Route che visitano un solo POI.

4.6. Note. Nelle prime versioni di Fill POI venivano generati solo inserimenti senza rimpiazzo.

Tuttavia, dopo un'analisi complessiva del risolutore, ci si é resi conto della necessitá di generare anche una varaiante di aggiunta con rimpiazzo. Questa esigenza nasce dal fatto che questo generatore é l'unico che attinge alla pool di POI non ancora visitati in alcuna Route.

I passi di analisi che hanno spinto l'introduzione della variante di aggiunta con rimpiazzo sono stati i seguenti:

(1) La soluzione iniziale contiene unicamente Route con un solo POI
(2) Nei primi step di risoluzione, pertanto, l'introduzione nelle Route esistenti di POI non ancora visitati risulta tendenzialmente molto appetibile, in virtú della bassa probabilitá di violazione dei vincoli di time-window; l'effetto é quindi il progressivo allungamento delle Route
(3) Le mosse di inserimento diventano quindi sempre meno appetibili: il risolutore inizia a selezionare mosse provenienti da altri generatori, che, per design, lavorano unicamente ricombinando le Route preesistenti.

L'effetto netto nell'avere inserimenti senza rimpiazzo é quindi che i POI coinvolti nel processo di risoluzione sono determinati troppo prematuramente, e che questa determinazione non viene mai riconsiderata.

L'introduzione di una variante di inserimento con rimpiazzo permette al risolutore di trasferire, in ogni momento, un POI non ancora visitato con uno giá visitato, senza pregiudicare il riempimento delle Route negli step iniziali di risoluzione. Il riempimento iniziale delle Route, oltre ad essere uno dei principali fattori di incremento del profitto della soluzione, é un processo sul quale gli altri generatori fanno affidamento. Essi sono, infatti, progettati interamente sul concetto di perturbazione e ricombinazione delle Route, che quindi devono essere caratterizzate da un certo grado di complessitá.

5. POI Move Generator

5.1. Scopo. Lo scopo di questo generatore é quello di scambiare o trasferire POI tra Route diverse. Pertanto, la generazione di mosse avviene soltanto se il numero di Route nella soluzione corrente é superiore a 1.

La generazione avviene selezionando le due Route che hanno la minore distanza, intendendo con distanza la minore distanza tra i POI che esse visitano.

Avendo selezionato le due Route, si generano, per ogni coppia di POI appartenenti rispettivamente alla prima e alla seconda Route, tutte le possibili varianti della mossa di scambio tra POI (Swap POI 1-1, Swap POI 0-1, Swap POI 1-0).

5.2. Mosse generate. Le mosse generate sono:

- Swap POI 1-1 (RouteA, PosA, RouteB, PosB): Scambia il POI che nella RouteA ha posizione PosA con il POI che nella RouteB ha posizione PosB.
- Swap POI 0-1 (RouteA, PosA, RouteB, PosB):
  - Elimina dalla RouteA il POI che ha posizione PosA
  - Inserisce il POI eliminato nella RouteB a posizione PosB
- Swap POI 1-0 (RouteA, PosA, RouteB, PosB):
  - Elimina dalla RouteB il POI che ha posizione PosB
  - Inserisce il POI eliminato nella RouteA a posizione PosA

5.3. Politica di generazione delle mosse. Per ogni coppia di Route A e B viene misurata la distanza minima tra i POI di A e i POI di B. La coppia di Route che ha la minor distanza minima viene selezionata per la generazione.

L'algoritmo di generazione é il seguente:

Algorithm 6 POI Move Generator

```
1:  function GENERATEPOIMOVES
2:      𝕄 ← ∅
3:      (routeA, routeB) = NEARESTROUTES()
4:      for all posA ∈ [0, size(routeA) − 1], posB ∈ [0, size(routeB) − 1] do
5:          𝕄 ← 𝕄 ∪ SWAPPOI(1_1, routeA, posA, routeB, posB)
6:          𝕄 ← 𝕄 ∪ SWAPPOI(0_1, routeB, posA, routeB, posB)
7:          𝕄 ← 𝕄 ∪ SWAPPOI(1_0, routeA, posA, routeB, posB)
8:      end for
9:      return 𝕄
10: end function
```

5.4. Esempio di generazione. Si riporta un esempio di scambio 1-0 che ha dato origine ad una nuova soluzione migliorante:

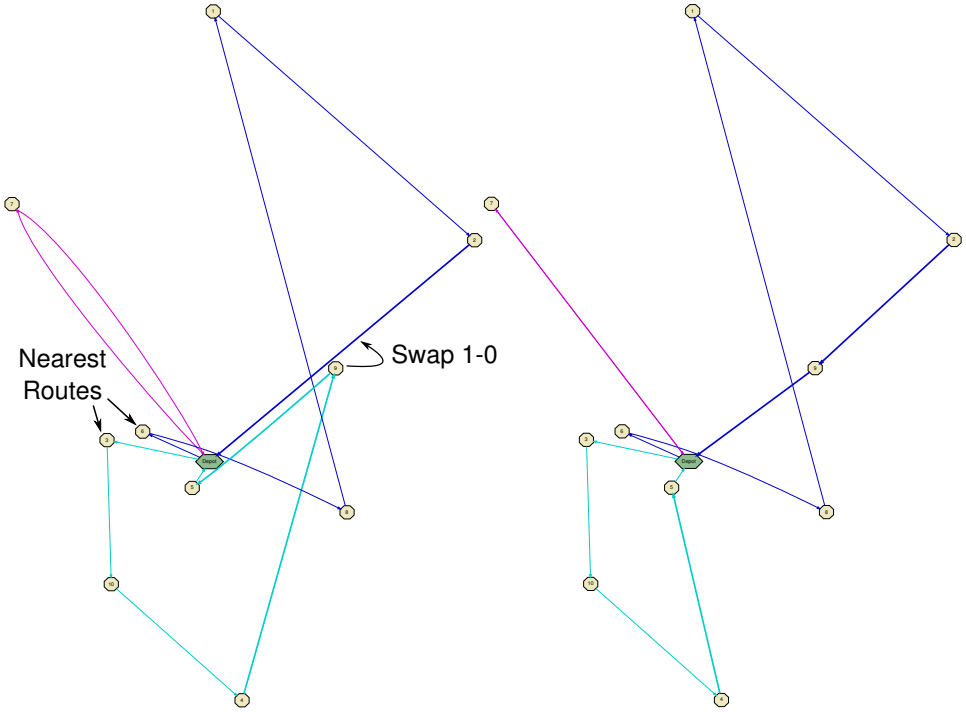


FIGURA 4. Swap POI 1-0 migliorante. Si nota come siano state selezionate le due Route di minore distanza (POI 3 visitato dal tour azzurro e POI 6 visitato dal tour blu). Successivamente sono state generate mosse di scambio in corrispondenza di tutte le posizioni. Il risolutore ha scelto come mossa migliore la cessione del POI 9 al tour di colore blu in corrispondenza dell'ultima posizione di visita.

5.5. Effetto tipico. A differenza di altri generatori, il POI Move Generator non dá origine a pattern immediatamente riconoscibili. Si puó affermare che questo é proprio il tratto distintivo di questo generatore: esso genera perturbazioni di piccola o media entitá, con un carico computazionale piuttosto contenuto e regolare, ma si é rivelato essenziale per la qualitá processo di risoluzione. Esso svolge, in maniera flessibile, varie funzioni:

- Talvolta funge da catalizzatore per gli altri generatori
- Nel caso di vuoti temporanei, ovvero nel caso in cui gli altri generatori abbiano temporaneamente difficoltá nel generare soluzioni (ad esempio a causa della tabu-list), il POI Move Generator puó aiutare il risolutore a spostarsi in aree piú fertili dello spazio delle soluzioni
- A volte funge da ottimizzatore, migliorando ad esempio quelle Route che sono appena state coinvolte da processi split-merge, che hanno maggior probabilitá di essere grossolane o inconsistenti

Per rendersi conto di quanto questo stadio sia essenziale, ecco l'esempio della risoluzione dello stesso problema due risolutori configurati esattamente nella, stessa maniera, con la sola differenza che nella prima run il POI Move Generator é stato disabilitato:

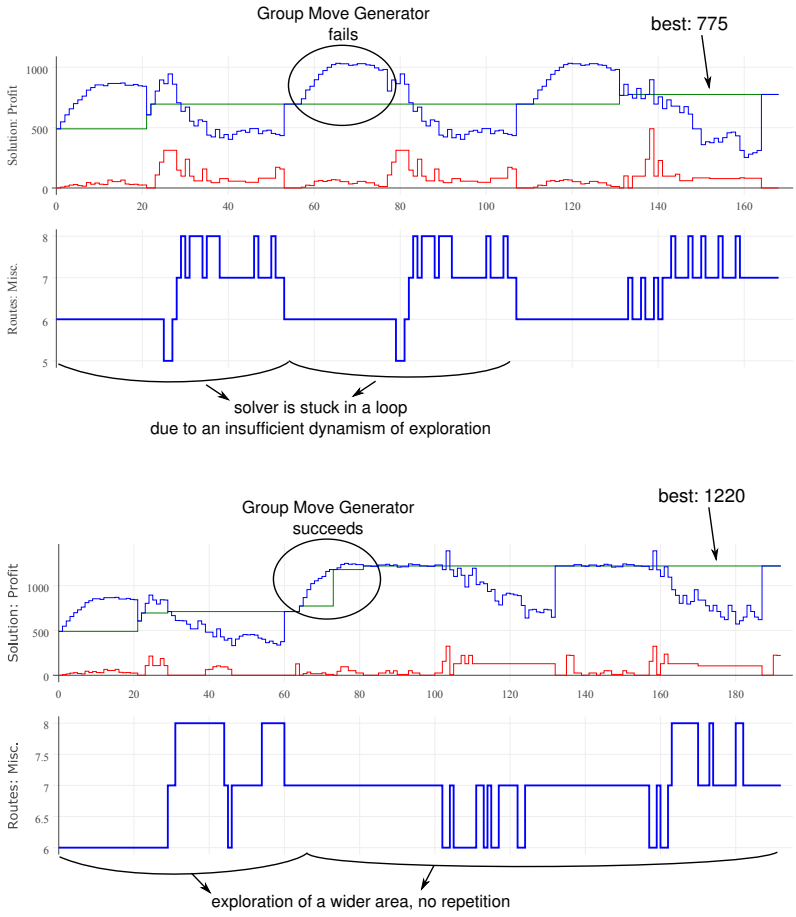


FIGURA 5. Andamento del risolutore con il POI Move Generator rispettivamente disabilitato ed abilitato.

6. Split/Merge Move Generator

6.1. Scopo. Lo scopo di questo generatore é di ricombinare Route esistenti mediante la suddivisione (split) di una Route o l'unione (merge) di due Route esistenti.

É interessante notare che nel risolutore, queste due mosse sono le uniche che esplicitamente cambiano il numero di Route in una soluzione. Le altre mosse che apportano questo tipo di cambiamento lo fanno in maniera marginale, ed inoltre possono soltanto ridurre il numero di Route:

- TransferGroup rimuove una Route se l'ultimo gruppo che conteneva viene trasferito
- SwapGroup 0-1 e 1-0 rimuovono una Route se essa contiene un solo POI e questo viene trasferito

La mossa di merge (Merge Routes) é pertanto essenziale per espandere l'utilizzo di Bus inattivi.

Questo tipo di operazioni rappresentano perturbazioni notevoli della soluzione corrente, e si sono rivelate molto difficili da mettere appunto. Un cambiamento anche di un'unitá nella dimensione massima di una pool puó cambiare completamente (e talvolta pregiudicare) il carattere del risolutore.

L'*operazione di suddivisione* agisce su una Route singola, pertanto viene selezionata una pool (di dimensione massima regolabile) di Route che soddisfano un particolare criterio, e, per ciascuna Route, viene generata un'operazione di split. Le Route candidate per l'operazione di split visitano almeno 2 POI e contengono almeno 2 Group.

L'*operazione di unione* agisce su una coppia di Route, pertanto viene selezionata una pool, di dimensione massima regolabile ≥ 2, contenente le Route che soddisfano il criterio di unione. Successivamente, per ciascuna coppia di Route nella pool, viene generata una mossa di merge.

6.2. Mosse generate. Le mosse generate sono:

- Split Route (RouteA, Bus):
  - Crea una nuova Route che impiega il Bus indicato
  - Ordina i Group della Route per profitto percepito decrescente e trasferisce quelli di ordine dispari alla nuova Route
  - Trasferisce i POI a posizioni dispari alla nuova Route
- Merge Routes (RouteA, RouteB):
  - Elimina le Route A e B
  - Crea una nuova Route
  - Aggiunge alla nuova Route i Group che erano nelle Route A e B
  - Iterativamente, aggiunge un POI alla volta scegliendo da quale Route attingere in maniera da mininmizzare, ad ogni iterazione, il timespan.

6.3. Politica di generazione delle mosse. La pool per le operazioni di split é determinata prediligendo le Route che hanno il timespan massimo.

La pool per le operazioni di merge é determinata invece prediligendo le Route che hanno il minimo prodotto $timespan * load\_ratio$.

L'algoritmo di generazione é il seguente:

Algorithm 7 Split/Merge Move Generator

Require: MAX_SPLIT ▷ max n. of Routes in split pool
Require: MAX_MERGE ▷ max n. of Routes in merge pool

function GENERATESPLITMERGEMOVES
  $\mathbb{M} \leftarrow \emptyset$
  $splitRoutes$ = SELECTSPLITROUTES()
  $mergeRoutes$ = SELECTMERGEROUTES()
  for all $route \in splitRoutes$ **do**
    $\mathbb{M} \leftarrow \mathbb{M} \cup$ SPLITROUTE($routeA$, 0)
  end for
  for all DISTINCT($routeA$, $routeB$) $\in mergeRoutes$ **do**
    $\mathbb{M} \leftarrow \mathbb{M} \cup$ MERGEROUTES($routeA$, $routeB$)
  end for
  return $\mathbb{M}$
end function

Dove le specifiche funzioni sono:

function SELECTSPLITROUTES(maxN)
  $routes$ = SORTBY($\mathbb{R}, r \mapsto timespan(r), ASC$)
  return TRIMTOSIZE($routes, maxN$)
end function
function SELECTMERGEROUTES(maxN)
  $routes$ = SORTBY($\mathbb{R}, r \mapsto load(r) * timespan(r), DESC$)
  return TRIMTOSIZE($routes, maxN$)
end function

6.4. Parametri.

MAX_SPLIT: Dimensione massima della pool di Route per l'operazione di split
MAX_MERGE: Dimensione massima della pool di Route per l'operazione di merge

6.5. Esempio di generazione. Ecco un esempio di mossa di split che ha determinato il rientro in regione ammissibile con una nuova soluzione ottima:

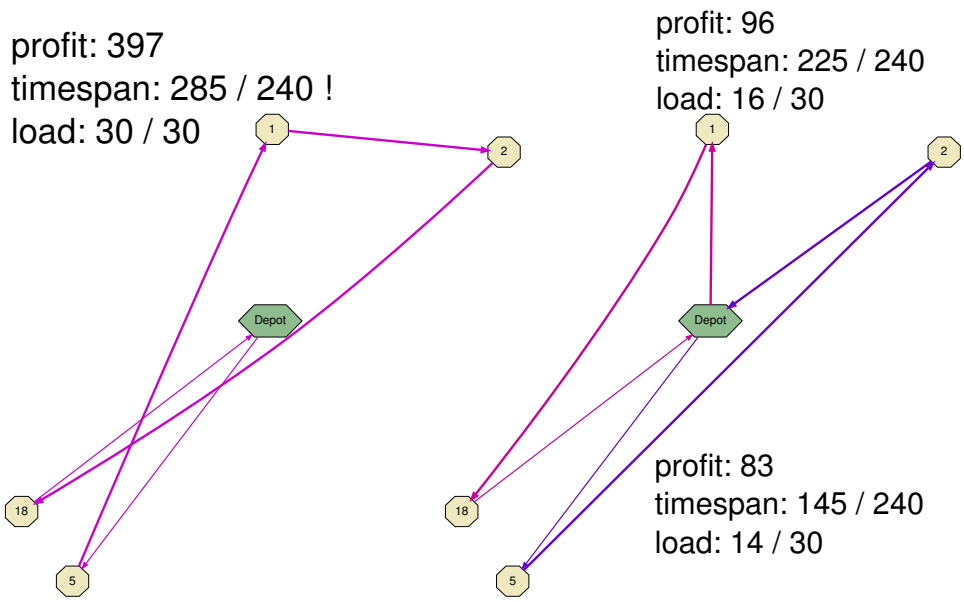


FIGURA 6. Mossa di Split migliorante. Il miglioramento é dovuto al fatto che si proveniva da una sequenza piuttosto greedy di mosse InsertPOI, che avevano dato origine ad una Route di grande profitto, ma ampiamente non ammissibile. La mossa di split ha ridotto il profitto di circa il 20%, ma ha ripristinato istantaneamente l'ammissibilitá, dando origine ad una nuova soluzione ottima.

Sebbene la mossa di Merge difficilmente dia origine ad un percorso ammissibile, in casi non patologici la soluzione risultante non é irrimediabilmente compromessa. Ad esempio, dopo questa mossa sono state eseguite dal risolutore, in sequenza:

- 5 mosse di Insert POI (con e senza replacement),
- 1 mossa di Split,
- 6 mosse di Insert POI (con e senza replacement),
- 1 mossa di Split

L'ultima mossa di Split ha dato origine ad una nuova soluzione ottima.

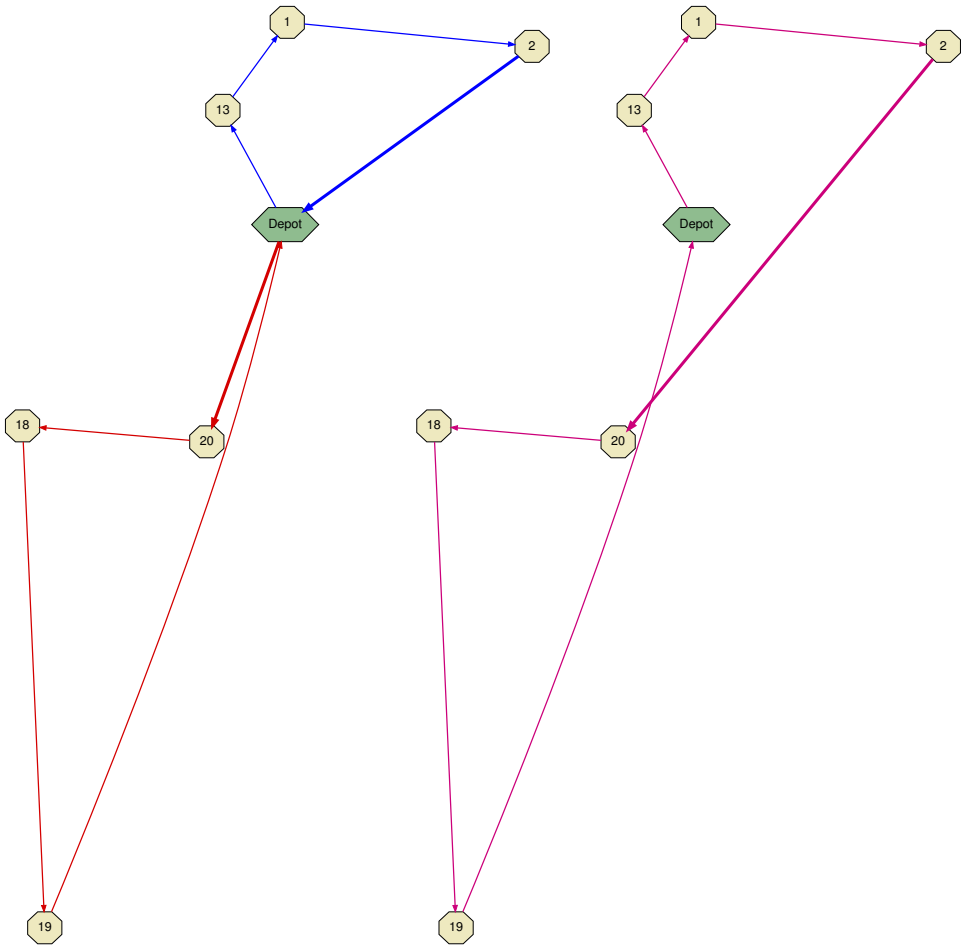


FIGURA 7. Mossa di Merge coinvolta nel percorso ad una nuova soluzione ottima. Sebbene visivamente il percorso sembri accettabile, esso in realtá é caratterizzato da una notevole violazione dei vincoli di time-window. Anche impiegando un'euristica che cerchi di ridurre localmente gli aumenti del timespan per portare a termine l'operazione di merge, é doveroso sottolineare come la messa appunto di questa mossa sia ancora una problematica aperta a miglioramenti.

6.6. Effetto tipico. L'effetto tipico del generatore é il seguente:

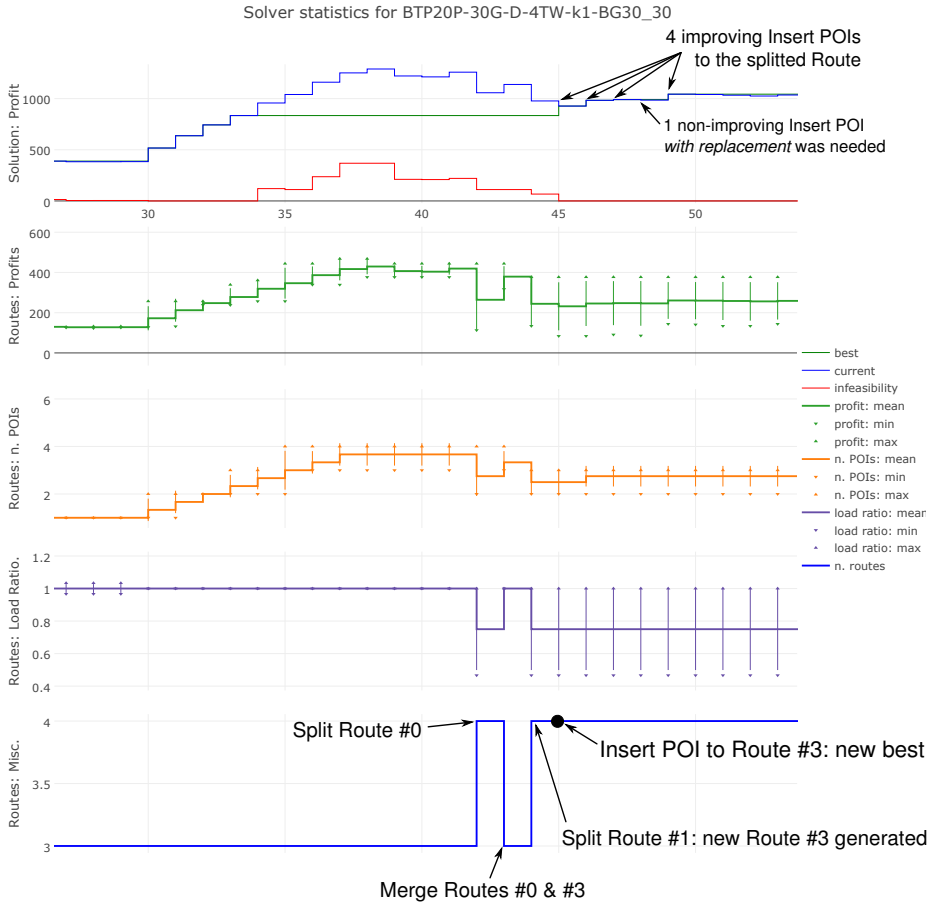


FIGURA 8. Sequenza di mosse miste provenienti da uno Split Merge Move Generator e da un POI Move Generator. L'effetto della ricombinazione delle route, seguite da mosse di scambio e trasferimento di POI, ha reso possibile il miglioramento della soluzione.

6.7. Note. Nelle versioni preliminari di questo generatore, anche la pool di Route candidate per l'operazione di split utilizzava come fattore di ordinamento il prodotto $timespan * load\_ratio$. Questo perché si intendeva comunque tentare di riunire Route scariche sia dal punto di vista del numero di POI visitati che dal punto di vista del carico di turisti.

A causa di questo, tuttavia si poteva osservare spesso il seguente comportamento indesiderato. La mossa di separazione tra Route (split) dá origine inevitabilmente ad una soluzione caratterizzata da un profitto minore: questo a causa del fatto che i gruppi di turisti, che inizialmente visitano un certo numero di POI, vengono separati in due Route, ciascuna delle quali contiene circa la metá dei POI iniziali. E' perció naturale che la soluzione peggiori da un punto di vista del mero profitto.

Questo comportava il fatto che l'algoritmo di ranking delle mosse scartasse molto spesso le mosse di tipo split, in favore delle mosse piú appetibili provenienti dagli altri generatori.

Per risolvere questo problema si é eliminato il fattore di carico dal prodotto, lasciando quindi come parametro di ordinamento delle Route il solo timespan. Questo permette a Route con pochi turisti, ma molti POI visitati, di essere suddivise: il fatto che esse abbiano pochi turisti si traduce in una minore entitá della diminuzione di profitto a fronte della mossa di suddivisione, con conseguente aumento della probabilitá che il risolutore la accetti come mossa migliore.

Un altro punto interessante degno di nota, che merita forse un'esplorazione ulteriore, é l'ispirazione al meccanismo biologico di suddivisione cellulare: in questo processo, una cellula si suddivide quando il rapporto fra massa (nel nostro caso rappresentato dal carico di turisti) e il volume (rappresentato dal timespan) raggiunge un valore soglia. L'applicazione di questo paradigma alla selezione delle Route candidate per le operazioni di split/merge resta un punto aperto ad esplorazioni future.

7. Opt1 Move Generator

7.1. Scopo. Lo scopo di questo generatore é di ottimizzare l'intero set di Route di una soluzione, attraverso la generazione di mosse di tipo Opt1, che, per costruzione, scambiano di posizione i POI all'interno di una Route.

7.2. Mosse generate. Le mosse generate sono:

- Opt1 (Route, PositionA, PositionB): Scambia i POI alle posizioni specificate.

7.3. Politica di generazione delle mosse. L'algoritmo di generazione é il seguente:

Algorithm 8 Opt1 Move Generator

```
1: function GENERATEOPT1MOVES
2:     M ← ∅
3:     for all route ∈ R do
4:         for all DISTINCT(posA, posB) ∈ [0, size(route) − 1] do
5:             M ← M ∪ OPT1(route, posA, posB)
6:         end for
7:     end for
8:     return M
9: end function
```

7.4. Parametri. Nessuno.

7.5. Effetto tipico. Quando questo generatore viene utilizzato in uno stadio standalone (modalitá scelta per il nostro algoritmo), il grafico delle statistiche non mostra miglioramenti. Questo fenomeno, spiegato meglio nelle note, é dovuto al fatto che una sequenza di mosse unicamente di tipo Opt1 é incapace di generare variazioni di profitti.

L'ottimizzazione delle Route avviene quindi solo sotto il punto di vista degli obiettivi secondari (riduzione del timespan complessivo): questo risulta evidente esaminando un esempio di intervento su una soluzione ottima operato da uno stadio che impiega un Opt1 Move Generator.

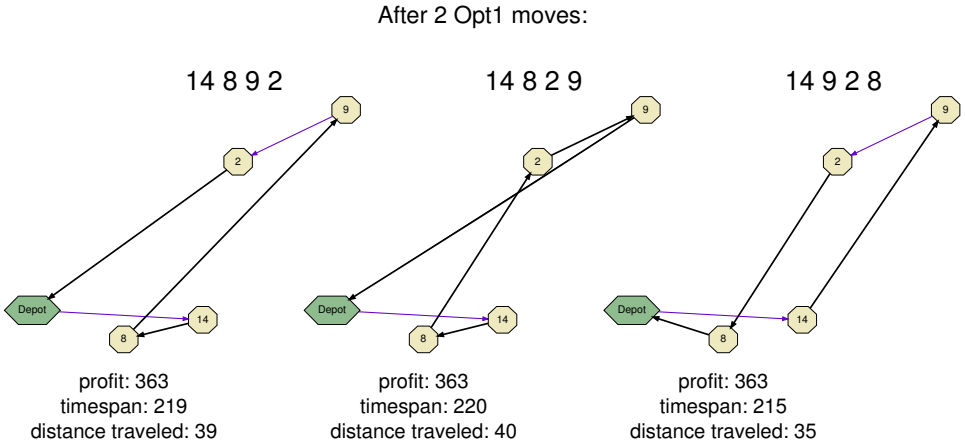


FIGURA 9. Sequenza di mosse Opt1 provenienti da un Opt1 Move Generator. Si nota come un numero ridotto di mosse sia sufficiente a dare origine ad una variazione notevole del percorso.

7.6. Note. Quando viene utilizzato in uno stadio a sé stante (come avviene nel nostro caso), questo generatore non puó produrre incrementi del profitto della soluzione: uno stadio, infatti, parte da una soluzione ammissibile iniziale e termina con la determinazione di una soluzione ottima ammissibile finale. Dato che lo scambio di POI all'interno di una Route non puó cambiare il profitto percepito dai gruppi di turisti che contiene, risulta evidente come tale incremento del profitto sia impossibile.

É comunque possibile utilizzare questo generatore in uno stadio che fa uso di altri generatori: in questo caso, le mosse Opt1 generate possono ripristinare la fattibilitá della soluzione, e quindi dare origine a nuove soluzioni strettamente miglioranti.

L'utilizzo in uno stadio stand-alone, come avviene nel nostro risolutore, é utile per migliorare gli obiettivi secondari al di lá del profitto: nel nostro caso, quello di diminuire il timespan totale della soluzione.

L'assenza di fattori di limitazione delle Route esaminate fa si che questo generatore possa diventare computazionalmente molto impegnativo, in caso di soluzioni con molte Route e molti POI visitati.

Parte 3

Risultati sperimentali

CAPITOLO 8

Istanze di problemi di benchmark

Per la parte sperimentale, é stato costruito un problem-set (denominato nel framework *problem set 2015*) costituito da 144 problemi di dimensione e complessitá variabile.

I problemi sono stati definiti combinando tra di loro elementi costruttivi di base: POI Files, Group Files, Profit Table Files, e Bus Group Files.

L'unico grado di libertá che resta fuori dalla definizione é il parametro k di massimo numero di visite ammesse per POI: per esso é specificabile un valore arbitrario > 0 al momento dell'inizializzazione dei problemi.

Gli esperimenti qui riportati utilizzano un parametro $k = 1$, che implica sostanzialmente la mutua esclusione tra le Route nella visita di un POI.

1. POI Files

Sono stati costruiti 4 grafi, contenenti rispettivamente 10, 20, 30 e 40 POI. Essi definiscono complessivamente 8 problemi considerando per ciascun grafo due diverse varianti di finestre temporali:

- la prima configurazione (detta *4-TW*) é pensata per piccoli tour con una durata massima di 240 unitá temporali;
- la seconda configurazione (detta *8-TW*) é pensata per tour lunghi con una durata massima di 480 unitá temporali.

Gli 8 POI Files (4 grafi $\times$ 2 configurazioni di time-window) assumono la forma di formulazioni VRP con time-window: per ogni vertice i sono indicate le coppie di coordinate (x, y), la finestra temporale $[et, lt]$ (risp. early time e latest time), ed il tempo di servizio st. Per il *depot* (POI 0), si ha $et = 0, lt = D, st = 0$ dove D rappresenta la massima durata consentita per una Route.

Per la distanza fra i POI viene utilizzata la semplice metrica euclidea intera:

$$\text{distance}\left(poi_1, poi_2\right) \triangleq \text{floor}\left(\sqrt{\left(poi_2.x - poi_1.x\right)^2 + \left(poi_2.y - poi_1.y\right)^2}\right)$$

Sono state considerate velocitá unitarie, quindi i valori della matrice dei tempi di percorrenza coincidono con quelli della matrice delle distanze.

2. Group Files, Bus Group Files

Sono stati generati due istanze di gruppi di turisti. La prima corrisponde ad un insieme di 30 gruppi di turisti, che si traducono complessivamente in 90 individui. La seconda istanza definisce 60 gruppi di turisti, che corrispondono a 180 individui.

Per quanto riguarda i Bus disponibili, sono stati generate 3 varianti di Bus, di capacitá rispettivamente di 15, 30 e 60 turisti ciascuno. Il numero di Bus disponibili é adattato all'istanza di gruppi di turisti utilizzata.

TABELLA 1. Bus disponibili.

Gruppi di turisti	Capacitá Bus		
	15	30	60
30	8	4	2
60	12	6	3

3. Profit Table Files

3.1. Profitti uniformi (PT-U). La classe di profitti uniformi é stata costruita nella maniera seguente: circa il 30% dei valori sono impostati a 0. Il resto dei valori sono stati impostati uniformemente in un intervallo predefinito.

Successivamente, la somma dei profitti per ciascun gruppo di turisti é stato riportato a 100, in modo da considerare le preferenze complessive di un gruppo piú importanti di altre.

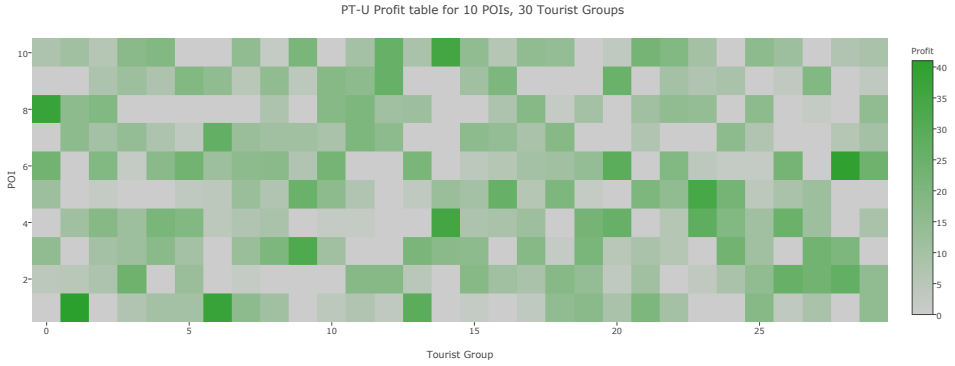


FIGURA 1. Heat map dei profitti di tipo U (uniform).

3.2. Profitti simili (PT-S). La classe di profitti simili é stata preparata generando inizialmente le preferenze del primo gruppo di turisti (in altre parole, la prima colonna della matrice).

Successivamente, il resto delle colonne sono state generate in maniera random in un vicinato della prima.

Ancora una volta, la somma dei profitti di ogni gruppo di turisti é stata portata a 100.

Questa classe di profitti riflette turisti con preferenze simili riguardo le visite.

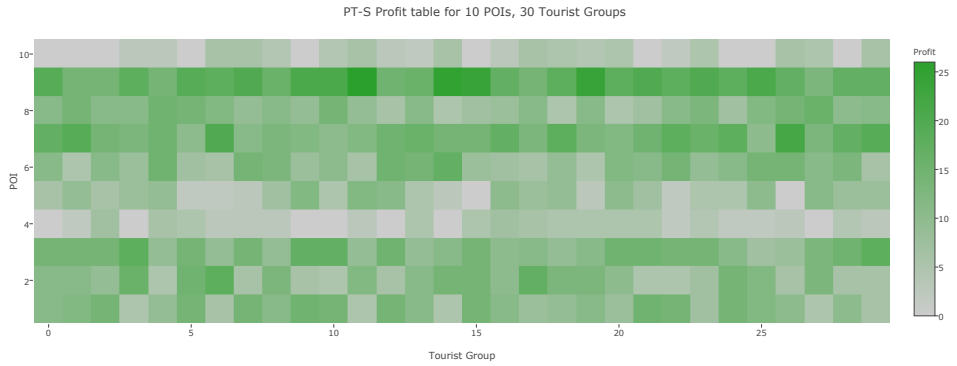


FIGURA 2. Heat map dei profitti di tipo S (similar). La similaritá dei profitti é riscontrabile visivamente dalla presenza di bande orizzontali.

3.3. Profitti dissimili (PT-D). La classe di profitti dissimili é stata preparata considerando inizialmente una funzione sinusoidale (sviluppata nella direzione dei gruppi di turisti), per quanto riguarda i profitti del primo POI. Ovviamente, la parte negativa della sinusoide é stata sostituita con un tratto identicamente nullo.

Successivamente, per il resto dei POI, i profitti sono stati generati considerando sfasamenti positivi della sinusoide.

Infine, la somma dei profitti di ogni gruppo é stata portata a 100.

Questa classe di profitti riflette turisti con diversi gusti riguardo le visite.

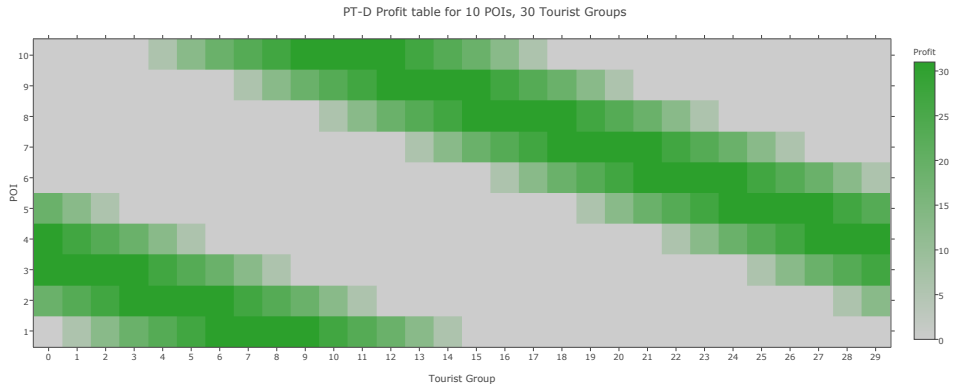


FIGURA 3. Heat map dei profitti di tipo D (dissimilar). Si nota l'andamento sinusoidale che si sviluppa sull'asse orizzontale e il progressivo sfasamento all'aumentare dell'indice del POI.

4. Conteggio Istanze

Il numero totale di istanze nel set di problemi di benchmark é quindi:

$$\text{POIFiles} \times \text{TW} \times \text{GroupFiles} \times \text{PT} \times \text{BusGroupFiles} =$$

$$4 \times 2 \times 2 \times 3 \times 3 =$$

$$= 144$$

Di queste 144 istanze, tuttavia, le varianti con 10 POIs, 60 Groups, Bus Group capacity 15 risultano impossibili se $k = 1$. L'assegnazione ottima di turisti ai Bus richiede 12 Bus, soddisfacendo il vincolo sul numero di Bus disponibili all'uguaglianza.

Tuttavia, se si considerano istanze con 10 POI e $k = 1$ risulta evidente che la visita esclusiva di 10 POI da parte di 12 Bus (Route) é impossibile.

Dato che i nostri esperimenti sono effettuati proprio con $k = 1$, queste istanze impossibili sono state scartate.

In conclusione, il numero di istanze di problemi da noi considerati si riduce a:

$$144 - \mathrm{TW} \times \mathrm{PT} =$$
$$144 - 6 =$$
$$138$$

CAPITOLO 9

Tuning dei parametri

Questa parte é dedicata alla messa appunto dell'algoritmo di risoluzione, attraverso la modifica dei parametri di configurazione.

Ci si prefigge l'obiettivo di regolare il risolutore per due diverse modalitá di funzionamento:

- La prima modalitá di funzionamento, detta *fast*, deve fornire risultati accettabili a fronte di un tempo di esecuzione inferiore al minuto.
- Nella seconda modalitá di funzionamento, detta *accurate*, si vuole prediligere la qualitá della soluzione ottenuta, senza vincolare esplicitamente il tempo di risoluzione, che comunque deve restare entro una soglia ragionevole (dell'ordine dei minuti).

Per guidare questo processo iterativo di messa appunto si considera un sottoinsieme campione dei problemi disponibili, limitato ma rappresentativo, che possa fornire riscontri sulle modifiche effettuate ad ogni iterazione in un tempo ragionevole.

TABELLA 1. Insieme campione per la fase di tuning.

Problem
10P 30G S 4TW k1 BG30
10P 30G D 4TW k1 BG30
10P 60G U 8TW k1 BG30
20P 30G S 4TW k1 BG15
20P 30G D 4TW k1 BG15
20P 60G U 8TW k1 BG30
30P 30G U 4TW k1 BG15
30P 30G S 8TW k1 BG60
40P 60G S 8TW k1 BG60

Si ritiene che questa fase sia essenziale, in quanto mette alla prova l'algoritmo e la sua implementazione, e permette di testare la risposta del risolutore ai parametri che ne dovrebbero influenzare il comportamento. É in questo punto che dovrebbero emergere eventuali problemi di funzionamento, problemi riguardanti la stabilitá, parametri superflui, ecc.

1. Modalitá fast

Innanzitutto, si parte dalla considerazione che un risolutore che lavora in modalitá veloce debba innanzitutto garantire qualche tipo di bound relativo ai passi di risoluzione effettuati, indipendentemente dalla difficoltá del problema affrontato.

Nel caso del nostro algoritmo, si ritiene che il primo punto di intervento debba essere la regolazione delle condizioni di terminazione.

La strategia infatti che intendiamo adottare per tenere sotto controllo il tempo di esecuzione si svolge in quattro passi, in cui, dopo una prima inizializzazione dei parametri di esplorazione con valori "neutrali", si svolgono due passi che riguardano proprio il controllo della terminazione.

In virtú del design multistadio adottato, i parametri per controllare la terminazione sono numerosi e offrono un ampio spazio di manovra. É infatti possibile stabilire la terminazione globale in termini dei parametri di terminazione dello stadio ripetitore di root-level:

(1) Il massimo numero di step complessivi;
(2) Il massimo numero di step senza miglioramenti - essendo a root-level questa condizione é controllata ad ogni passo, qualunque sia il sotto-stadio attualmente in esecuzione;
(3) Il massimo numero di riavvolgimenti.

É inoltre possibile fermare l'esecuzione dei sotto-stadi (Group, POI, Opt) attraverso le relative condizioni di terminazione:

(1) Il massimo numero di step complessivi;
(2) Il massimo numero di step senza miglioramenti

Queste ultime condizioni non arrestano direttamente il risolutore ma provocano il passaggio allo stadio risolutivo successivo. Esse permettono quindi un controllo indiretto sulla terminazione dello stadio root-level.

Strategia

In sintesi, la strategia che si intende impiegare consiste nel:

(1) Inizializzare i parametri che riguardano l'esplorazione con valori "neutrali";
(2) Impostare i limiti hard sul numero di passi effettuati, agendo sullo stadio di root-level;
(3) Regolare i limiti sul numero di passi compiuti da ciascun stadio;
(4) Espandere eventualmente l'impiego di risorse computazionali da parte dei generatori di mosse fino al raggiungimento dei vincoli temporali.
(5) Regolare i parametri di esplorazione, ovvero i parametri che controllano la modulazione $alpha$ e grado persistenza in tabu-list.

1.1. Primo passo: Init dei parametri di esplorazione. Procediamo quindi impostando i parametri di esplorazione a valori neutrali.

Impostiamo il coefficiente di penalizzazione α al valore neutrale 1.0. In questo modo otteniamo una penalizzazione del profitto pari all'indice di non-ammissibilitá.

Decidiamo di disabilitare temporaneamente la modulazione α, in quanto essa non influenza direttamente i tempi di esecuzione. Per far ció é sufficiente impostare la soglia β di modulazione ad un valore molto grande.

Il parametro θ controlla il numero di step di persistenza di una soluzione all'interno della tabu-list. Dato che la persistenza di una soluzione in tabu-list comporta un carico computazionale, stimiamo un upper-bound di questo consumo mantenendo una completa persistenza, fissando $theta$ ad un valore molto grande.

Si stabiliscono quindi i seguenti parametri esplorativi.

TABELLA 3. Parametri di esplorazione neutri.

Stadio	Parametro	Valore
Stadio 1 (Groups)	α	1.0
	β	10^6
	θ	10^6
Stadio 2 (Routes)	α	1.0
	β	10^6
	θ	10^6
Stadio 3 (Opt1)	α	1.0
	β	10^6
	θ	10^6

1.2. Secondo passo: Condizioni di terminazione, stadio di root-level. Per impostare il limite globale sul numero di iterazioni, si intende eseguire un numero crescente di iterazioni su un problema di grandi dimensioni osservando l'andamento dei tempi di esecuzione.

É importante che in questo esperimento la distribuzione del carico computazionale tra i vari stadi sia realistica, perció si adotta la seguente politica:

- Si disabilita la terminazione prematura di uno stadio, attraverso l'impostazione del massimo numero di passi consentiti senza miglioramenti ad un valore molto alto;
- Si imposta il massimo numero di rewind consentito ad un valore molto alto, per evitare terminazioni premature dello stadio di root-level;
- Si imposta il massimo numero di step consentiti per ciascuno stadio con valori piccoli ma proporzionati.

Questa politica dá origine ai seguenti parametri, indicando con x il valore di test per il massimo numero di iterazioni:

TABELLA 4. Valori sonda per determinare il massimo numero di iterazioni.

Stadio	Parametro	Valore
Root-Level	Max steps	**x**
	Max steps no best	10^6
	Max rewind	10^6
Stadio 1 (Groups)	Max steps	5
	Max steps no best	10^6
Stadio 2 (Routes)	Max steps	20
	Max steps no best	10^6
Stadio 3 (Opt1)	Max steps	1
	Max steps no best	10^6

Giunti a 250 step, si ottengono tempi di circa 60 secondi, nell'ordine quindi dei tempi di esecuzione desiderati per la modalitá fast.

TABELLA 5. Tempo di esecuzione per $x = 250$ steps, problema di grandi dimensioni.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
40P 60G S 8TW k1 BG60	262	1100	+838 (≈ 320%)	250	188	45	≈ **66**

Utilizziamo questo valore come limite hard al numero di iterazioni dello stadio root-level. Con un numero cosí basso di step massimi, la specifica di un numero massimo di step senza miglioramenti viene giudicata rischiosa: una terminazione prematura di questo tipo potrebbe pregiudicare un'eventuale soluzione migliorante a fronte di un tempo di calcolo inferiore di qualche secondo. Viene perció stabilito per questo parametro un margine ampio di 100 step, in modo da cortocircuitare la risoluzione solo in casi estremi.

La stessa considerazione vale per quanto riguarda il massimo numero consentito di riavvolgimenti: non si vuole far terminare il risolutore a causa di questo. Infatti, nell'eventualitá che gli stadi non diano origine a miglioramenti, entrerebbe comunque in funzione il valore limite di 100 step senza miglioramenti. É comunque opportuno stabilire un valore ragionevole per questo parametro. Considerando 3 stadi risolutivi, ciascuno dei quali ha indicativamente bisogno di almeno 20 passi per dare origine a risultati, ed un limite globale di 250 passi, puó essere stimato un numero indicativo di riavvolgimenti massimi raggiungibili:

$$250 / (3 \times 20) \approx 4$$

Si ottengono quindi i seguenti parametri.

TABELLA 7. Valori determinati per i parametri di terminazione dello stadio di root-level.

Stadio	Parametro	Valore
Root-Level	Max steps	**250**
	Max steps no best	**100**
	Max rewind	**4**
Stadio 1 (Groups)	Max steps	5
	Max steps no best	10^6
Stadio 2 (Routes)	Max steps	20
	Max steps no best	10^6
Stadio 3 (Opt1)	Max steps	1
	Max steps no best	10^6

1.2.1. *Fine secondo passo: controllo.* É opportuno controllare ora l'andamento dei tempi di esecuzione per l'insieme di test, trascurando la qualitá delle soluzioni ottenute.

TABELLA 8. Termine passo 1: controllo del rispetto del bound temporale.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	686	+219 (≈ 47%)	250	64	10	≈ **5**
10P 30G D 4TW k1 BG30	506	1625	+1119 (≈ 221%)	250	65	10	≈ **5**
10P 60G U 8TW k1 BG30	882	1738	+856 (≈ 97%)	250	32	8	≈ **7**
20P 30G S 4TW k1 BG15	285	398	+113 (≈ 40%)	250	31	5	≈ **16**
20P 30G D 4TW k1 BG15	269	533	+264 (≈ 98%)	250	11	7	≈ **10**
20P 60G U 8TW k1 BG30	454	474	+20 (≈ 4%)	250	1	2	≈ **20**
30P 30G U 4TW k1 BG15	173	336	+163 (≈ 94%)	250	39	8	≈ **36**
30P 30G S 8TW k1 BG60	195	1025	+830 (≈ 426%)	250	130	28	≈ **50**
40P 60G S 8TW k1 BG60	262	1100	+838 (≈ 320%)	250	188	45	≈ **66**

I vincoli temporali sono soddisfatti. Si riscontra inoltre che i tempi di risoluzione rispecchiano la difficoltá e la dimensione del problema, come é lecito supporre.

1.3. Terzo passo: condizione di terminazione, sotto-stadi. A questo punto si dispone di una limitazione globale sul numero di iterazioni eseguite. In questo passo si cercano di distribuire le iterazioni disponibili tra i diversi stadi, in maniera ragionata.

A differenza di quanto considerato per il primo passo di tuning, la specifica sulla terminazione in base al numero di passi effettuati senza miglioramenti risulta in questo caso molto conveniente. Mediante questa, infatti, si scoraggia l'utilizzo vano dei passi risolutivi di cui si dispone; questo avviene inoltre in maniera adattiva, a differenza del limite sul numero di step complessivi che agisce invece in maniera rigida.

Pertanto l'intento ora é di esaminare nello specifico gli stadi a disposizione e di determinare per ciascuno di essi il numero di passi dopo i quali smettono di fornire miglioramenti, e, a partire da questo valore, stabilire anche un numero massimo di step consentiti.

Terzo Stadio

Il terzo stadio é uno stadio di ottimizzazione fine: esso si basa unicamente su un generatore Opt1 e, come notato nella sezione 7, esso non migliora il profitto della soluzione. Essendo in un contesto che richiede soluzioni in tempi brevi, risulta evidente che l'impiego di questo stadio deve essere ristretto notevolmente. Si stabilisce pertanto che lo stadio termini a fronte di 4 consecutive mosse non miglioranti. Per fissare anche un numero massimo di passi consentiti, si stima il numero indicativo di mosse Opt1 desiderate.

Considerando il numero POI visitati dalle Route piú lunghe, si osserva che: un problema con molti POI, finestre temporali permissive, e basso grado di contesa dei POI é del tipo 40P-60G-S-8TW-k1-BG60_60 (presente nell'insieme campione); per esso, le Route lunghe ammissibili visitano al massimo 10 POI. Si osserva inoltre che il massimo numero

di Bus a disposizione all'interno di tutto il problem set é di 12. In questo senso, ipotizzando di scambiare un terzo dei POI per ciascuna Route richiederebbe $\frac{10}{3} \times 12 = 40$ step. Considerando infine che il numero di volte che questo stadio puó entrare in funzione é pari al numero massimo di rewind (fissato a 4), ed ipotizzando di distribuire tali scambi all'interno dell'esecuzione complessiva del risolutore, otteniamo $40/4 \approx 10$ come numero di step massimi consentiti.

Primo Stadio

Si considera ora il primo stadio. Esso si basa unicamente sul generatore Group Move, ed ha il compito di equalizzare il carico ed il profitto tra le diverse Route attraverso lo spostamento di gruppi di turisti. Per la maniera in cui opera il generatore sottostante (descritta nella sezione 3), si puó osservare che gli eventuali miglioramenti della soluzione avvengono per lo piú in una corta sequenza iniziale, con l'eccezione di qualche sporadico miglioramento dopo l'uscita temporanea dalla regione di ammissibilitá. Misurando le distanze tra i miglioramenti si ottiene un valore di approssimativamente 10 step; lo step a cui avviene l'ultimo miglioramento é normalmente inferiore al 20; la catena iniziale di miglioramenti é normalmente di lunghezza inferiore ai 20 passi. Viene quindi scelto 15 come valore adeguato di numero massimo di step consecutivi senza miglioramenti e 50 come limite ultimo al massimo numero di step.

Secondo Stadio

Il secondo stadio é quello piú oneroso in termini computazionali, in quanto esso fa uso di tutti i generatori di mosse che agiscono sulle Route (ad eccezione del generatore di mosse Opt1). Viene innanzitutto riservato il massimo numero di step consentiti, ottenuto dai corrispondenti vincoli assegnati agli altri stadi:

$$\underbrace{50}_{\text{Stadio 1}} + \underbrace{x}_{\text{Stadio 2}} + \underbrace{10}_{\text{Stadio 3}} = 250$$
$$\implies x = 190$$

La stima del massimo numero di step consentiti al secondo stadio risulta molto difficile. Il valore agisce da discriminante tra un'esecuzione lunga con pochi rewind, ed un'esecuzione breve ma molte ripetizioni degli stadi. La lunghezza dell'esecuzione di questo stadio inoltre influenza il grado di uscita dalla regione ammissibile che é possibile ottenere: la terminazione troppo prematura puó tradursi nell'impossibilitá da parte dello stadio di riportarsi in regione ammissibile. A questo punto é prematuro pronunciarsi su questa scelta: si fissa quindi un valore intermedio, ottenuto rapportando il corrispondente valore fissato per il primo stadio (15) con il lavoro richiesto al secondo. Si opta quindi per un valore pari a 35.

Si riportano quindi i parametri di terminazione determinati.

TABELLA 10. Parametri di terminazione.

Stadio	Parametro	Valore
Root-Level	Max steps	250
	Max steps no best	100
	Max rewind	4
Stadio 1 (Groups)	Max steps	**50**
	Max steps no best	**15**
Stadio 2 (Routes)	Max steps	**190**
	Max steps no best	**35**
Stadio 3 (Opt1)	Max steps	**10**
	Max steps no best	**4**

1.3.1. *Fine terzo passo: controllo.* Prima di aggiustare i parametri che riguardano lo stato tabu ed i generatori, é opportuno avere un riscontro su quanto stabilito finora.

TABELLA 11. Riscontro dopo la distribuzione delle risorse di calcolo attraverso il controllo delle condizioni di terminazione.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	676	+209 (≈ 45%)	176	76	13	≈ 6
10P 30G D 4TW k1 BG30	506	1013	+507 (≈ 100%)	124	24	4	≈ 4
10P 60G U 8TW k1 BG30	882	1902	+1020 (≈ 116%)	178	78	11	≈ 8
20P 30G S 4TW k1 BG15	285	398	+113 (≈ 40%)	154	54	5	≈ 8
20P 30G D 4TW k1 BG15	269	533	+264 (≈ 98%)	121	21	7	≈ 5
20P 60G U 8TW k1 BG30	454	474	+20 (≈ 4%)	101	1	2	≈ 9
30P 30G U 4TW k1 BG15	173	336	+163 (≈ 94%)	213	113	10	≈ 28
30P 30G S 8TW k1 BG60	195	1177	+982 (≈ 504%)	250	157	33	≈ 36
40P 60G S 8TW k1 BG60	262	1007	+745 (≈ 284%)	197	97	39	≈ 45

I tempi di esecuzione restano nei limiti. Si nota inoltre che in molti casi la terminazione veloce ha comportato un risparmio di tempi di calcolo inutili. É opportuno che la terminazione non avvenga negli step immediatamente successivi allo step in cui si é trovata l'ultima soluzione migliorante: questo infatti comporterebbe il rischio di una terminazione prematura.

1.4. Quarto passo: espansione delle risorse attribuite ai generatori. In questo passo aumentiamo le risorse a disposizione dei generatori di mosse fino a raggiungere i vincoli temporali imposti dalla modalitá fast.

Tutti i parametri corrispondono a dimensioni massime di pool contenenti gruppi di turisti o Route. I Move Generators (M.G.) attingono da queste pool le possibili entitá coinvolte nelle mosse che vengono generate.

Riportiamo innanzitutto i valori utilizzati finora, che sono stati determinati empiricamente in fase di sviluppo dei generatori: i parametri assumono valori pari ad una frazione del numero di Bus e gruppi di turisti presenti nei problemi di esempio.

TABELLA 13. Elenco dei parametri di tuning per i generatori di mosse.

Stadio	Generatore Impiegato	Parametro	Valore
Stadio 1 (Groups)	Group M.G.	Max Low-Profit Routes	3
		Max High-Load Routes	3
		Max Unsatisfied Groups	8
Stadio 2 (Routes)	FillPOI M.G.	Max Nearest POIs	8
		Max Highly-Profitable Routes	4
		Max Low-Timespan Routes	8
	SplitMerge M.G.	Max Split-Pool Routes	4
		Max Merge-Pool Routes	4
	SwapPOI M.G.	*none*	
Stadio 3 (Opt1)	Opt1 M.G.	*none*	

Ripristino di θ

Prima di procedere, riduciamo il carico computazionale introdotto artificialmente dovuto al parametro θ di persistenza delle soluzioni in tabu-list, rapportando tale valore al parametro sul massimo numero di step consentiti senza miglioramenti.

TABELLA 14. Parametri di esplorazione: eliminazione del carico artificiale sulla tabu-list (parametro θ).

Stadio	Parametro	Valore
Stadio 1 (Groups)	α	1.0
	β	10^6
	θ	**4**
Stadio 2 (Routes)	α	1.0
	β	10^6
	θ	**8**
Stadio 3 (Opt1)	α	1.0
	β	10^6
	θ	**1**

Una maniera naturale di fornire piú risorse ai generatori mantenendo l'equilibrio é aumentando il valore Max Nearest POIs, che si discosta molto dal numero massimo di POI presenti nelle istanze piú grandi. Inoltre aumentiamo il numero massimo di Route nella pool Low-Profit.

TABELLA 15. Primo tentativo di tuning.

Stadio	Generatore Impiegato	Parametro	Valore
Stadio 1 (Groups)	Group M.G.	**Max Low-Profit Routes**	~~3~~ **6**
		Max High-Load Routes	3
		Max Unsatisfied Groups	8
Stadio 2 (Routes)	FillPOI M.G.	**Max Nearest POIs**	~~8~~ **16**
		Max Highly-Profitable Routes	4
		Max Low-Timespan Routes	8
	SplitMerge M.G.	Max Split-Pool Routes	4
		Max Merge-Pool Routes	4
	SwapPOI M.G.	*none*	
Stadio 3 (Opt1)	Opt1 M.G.	*none*	

Si ottengono i seguenti risultati.

TABELLA 16. Limiti temporali raggiunti dopo l'aumento delle risorse fornite ai generatori di mosse.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	676	+209 (≈ 45%)	176	76	13	≈ 4
10P 30G D 4TW k1 BG30	506	1013	+507 (≈ 100%)	124	24	4	≈ 3
10P 60G U 8TW k1 BG30	882	1703	+821 (≈ 93%)	221	121	25	≈ 30
20P 30G S 4TW k1 BG15	285	435	+150 (≈ 53%)	171	71	12	≈ 20
20P 30G D 4TW k1 BG15	269	503	+234 (≈ 87%)	120	20	6	≈ 12
20P 60G U 8TW k1 BG30	454	1688	+1234 (≈ 272%)	250	223	48	≈ **73**
30P 30G U 4TW k1 BG15	173	534	+361 (≈ 209%)	250	213	26	≈ 45
30P 30G S 8TW k1 BG60	195	968	+773 (≈ 396%)	177	77	24	≈ 30
40P 60G S 8TW k1 BG60	262	1374	+1112 (≈ 424%)	230	130	33	≈ **69**

Ci si é portati al limite della finestra temporale concessa per la modalitá fast di risoluzione.

1.4.1. *Quarto passo: uno sguardo ai risultati.* É giunto il momento di gettare uno sguardo anche ai progressi in merito alla qualitá delle soluzioni ottenute. Innanzitutto, grazie alle ultime modifiche é stata sbloccata la situazione critica rappresentata dal problema 20P-60G-U-8TW-k1-BG30_30. Per esso, infatti veniva trovata una sola soluzione migliorante al primo passo risolutivo, che determinava un arresto al passo 101 per assenza di miglioramenti nei 100 step successivi. Ora si ottengono 48 miglioramenti, dei quali l'ultimo avviene al passo 223, giungendo alla terminazione a causa del limite hard di 250 passi. Questo lascia intravedere che esiste un margine di miglioramento per la modalitá accurate.

TABELLA 18. Sblocco della situazione critica per il problema 20P-60G-U-8TW-k1-BG30_30.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
20P 60G U 8TW k1 BG30	454	474	+20 ($\approx$ 4%)	101	1	2	$\approx$ 9
20P 60G U 8TW k1 BG30	454	**1688**	+1234 ($\approx$ 272%)	250	**223**	**48**	$\approx$ 73

É interessante indagare sulla causa del miglioramento. Per questo si riportano i grafici statistici delle due risoluzioni.

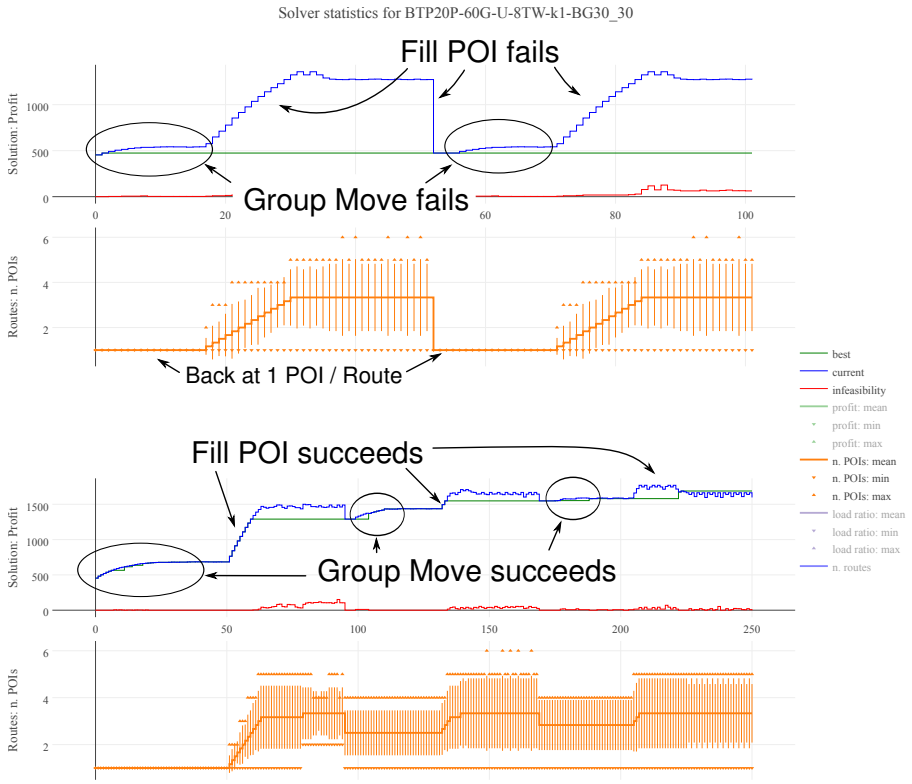


FIGURA 1. Statistiche di risoluzione per 20P-60G-U-8TW-k1-BG30_30 prima e dopo le modifiche apportate nel quarto passo. L'aumento delle dimensioni massime delle pool considerate dai generatori FillPOI e GroupMove ha determinato il raggiungimento di soluzioni miglioranti.

In linea di massima, si nota un'equalizzazione complessiva delle performance del risolutore, con qualche peggioramento e numerosi miglioramenti, in entrambe i casi (trascurando l'evento isolato appena citato) di lieve entitá.

1.5. Quinto passo: parametri di esplorazione. L'obiettivo per questo ultimo passo é di aggiustare i parametri che controllano l'esplorazione, cercando di migliorare quanto possibile la qualitá delle soluzioni.

Il fattore principale che verrá influenzato riguarda principalmente la cosiddetta modulazione α, ovvero la commutazione del risolutore fra le fasi di intensificazione e diversificazione. La considerazione di fondo riguardo l'innesco di queste commutazioni riguarda la relazione che si vuole stabilire fra esse e quella parte di dinamica di esplorazione dettata dalla tabu-list. L'uscita di una soluzione dalla tabu-list é sostanzialmente la concessione al risolutore del permesso di riconsiderare il percorso da intraprendere e "ritornare sui propri passi". In relazione a questo, l'innesco di una fase di diversificazione rappresenta l'esatto opposto, ovvero lo stimolo ad abbandonare l'area correntemente sotto esame attraverso l'uscita dalla regione di ammissibilitá.

Relazione tra Tabu List e Modulazione α

É chiaro quindi che, per ottenere una buona dinamica di esplorazione, occorre far si che sia possibile questa scelta tra la diversificazione ed il ritorno su soluzioni giá esplorate. Questo richiede che il parametro β di innesco della modulazione α assuma valori molto

vicini a quelli assunti dal parametro θ di persistenza in tabu-list. La scelta tra $\beta > \theta$ e $\beta < \theta$ consente di fornire un suggerimento, o *hint* al risolutore su quale politica prediligere:

$\beta < \theta$: la diversificazione si innesca prima dell'uscita della soluzione dalla tabu-list: Bias sulla diversificazione.

$\beta > \theta$: la soluzione esce dalla tabu-list, dopodiché si attiva eventualmente la diversificazione: Bias sul backtracking.

$\beta = \theta$: assenza di bias nella la scelta fra diversificazione e backtracking.

Terzo Stadio

1.5.1. *Interventi su β.* Per quanto riguarda il terzo stadio (Opt1), é stato impostato $\beta \equiv \theta = 1$. Tuttavia, questa modifica non ha comportato nell'insieme campione alcun innesco di processi di diversificazione: questo probabilmente é dovuto al numero molto basso di passi compiuti (≤ 10).

Primo Stadio

Per il primo stadio (Groups), é stato osservato che per problemi con pochi gruppi (di tipo 30G), il picco di qualitá si ha con $\beta \equiv \theta$. L'hint $\beta > \theta$ comporta miglioramenti per problemi con molti gruppi (60G), ma risulta in generale una scelta peggiore a causa dell'entitá dei peggioramenti introdotti. L'impostazione $\beta < \theta$ genera sempre soluzioni uguali o peggiori rispetto agli altri casi.

TABELLA 20. Risultati per differenti valori di β per il primo stadio.

Problem	Improvement		
	$\beta = 2 < \theta = 4$	$\beta \equiv \theta = 4$	$\beta = 8 > \theta = 4$
10P 30G S 4TW k1 BG30	+263 ($\approx$ 56%)	+614 ($\approx$ **131%**)	+209 ($\approx$ 45%)
10P 30G D 4TW k1 BG30	+507 ($\approx$ 100%)	+507 ($\approx$ 100%)	+507 ($\approx$ 100%)
10P 60G U 8TW k1 BG30	+986 ($\approx$ 112%)	+1213 ($\approx$ **138%**)	+1244 ($\approx$ **141%**)
20P 30G S 4TW k1 BG15	+129 ($\approx$ 45%)	+267 ($\approx$ **94%**)	+150 ($\approx$ 53%)
20P 30G D 4TW k1 BG15	+453 ($\approx$ **168%**)	+363 ($\approx$ 135%)	+455 ($\approx$ **169%**)
20P 60G U 8TW k1 BG30	+422 ($\approx$ 93%)	+748 ($\approx$ 165%)	+1267 ($\approx$ **279%**)
30P 30G U 4TW k1 BG15	+197 ($\approx$ 114%)	+331 ($\approx$ **191%**)	+197 ($\approx$ 114%)
30P 30G S 8TW k1 BG60	+776 ($\approx$ 398%)	+776 ($\approx$ 398%)	+776 ($\approx$ 398%)
40P 60G S 8TW k1 BG60	+139 ($\approx$ 53%)	+968 ($\approx$ **369%**)	+968 ($\approx$ **369%**)

Si decide quindi di impostare per il primo stadio β a valore 4.

Secondo Stadio

Per quanto riguarda il secondo stadio (Routes) eseguiamo lo stesso tipo di esperimento, con $\beta \in \{5, 8, 12\}$.

TABELLA 22. Risultati per differenti valori di β per il secondo stadio.

Problem	Improvement		
	$\beta = 5 < \theta = 8$	$\beta \equiv \theta = 8$	$\beta = 12 > \theta = 8$
10P 30G S 4TW k1 BG30	+616 ($\approx$ 132%)	+614 ($\approx$ 131%)	+614 ($\approx$ 131%)
10P 30G D 4TW k1 BG30	+1119 ($\approx$ **221%**)	+507 ($\approx$ 100%)	+507 ($\approx$ 100%)
10P 60G U 8TW k1 BG30	+1213 ($\approx$ 138%)	+1213 ($\approx$ 138%)	+1213 ($\approx$ 138%)
20P 30G S 4TW k1 BG15	+322 ($\approx$ **113%**)	+299 ($\approx$ 105%)	+314 ($\approx$ **110%**)
20P 30G D 4TW k1 BG15	+718 ($\approx$ **267%**)	+632 ($\approx$ 235%)	+697 ($\approx$ 259%)
20P 60G U 8TW k1 BG30	+748 ($\approx$ 165%)	+748 ($\approx$ 165%)	+748 ($\approx$ 165%)
30P 30G U 4TW k1 BG15	+390 ($\approx$ 225%)	+390 ($\approx$ 225%)	+382 ($\approx$ 221%)
30P 30G S 8TW k1 BG60	+1025 ($\approx$ **526%**)	+927 ($\approx$ 475%)	+891 ($\approx$ 457%)
40P 60G S 8TW k1 BG60	+1066 ($\approx$ **407%**)	+1035 ($\approx$ 395%)	+1036 ($\approx$ 395%)

Dagli esperimenti emerge chiaramente che l'introduzione della modulazione *alpha* porta sostanziali miglioramenti al processo di esplorazione e alla qualitá delle soluzioni. Emerge inoltre che per questo passo é tendenzialmente piú vantaggioso attuare una politica di diversificazione piuttosto che ricorrere al backtracking. Questa differenza rispetto al primo stadio é probabilmente dettata da una maggiore complessitá delle mosse attuate: l'uscita dalla regione ammissibile consente alle profonde modifiche apportate ad esempio dalle mosse split/merge di essere temporaneamente accettate e di mettere a frutto poi il loro potenziale. Seguiamo perció i risultati fornendo l'hint verso la diversificazione, attraverso la scelta $\beta = 5 < \theta$.

1.5.2. *Interventi sul valore iniziale di α.* L'ultimo parametro da fissare é quello che stabilisce il valore iniziale del fattore di modulazione α. Questo parametro influenza il grado di inammissibilitá delle soluzioni che lo stadio é disposto ad accettare nelle fasi iniziali della risoluzione. Dopo questa fase iniziale entra comunque in funzione il meccanismo di modulazione, che agisce reattivamente sul valore di α. Questo puó portare a pensare che il valore iniziale sia quindi superfluo. In realtá esso influenza molto la qualitá delle soluzioni, soprattutto nei casi - come il nostro - in cui lo stadio esegua un numero di iterazioni ridotto. Questa influenza risulta ancora piú marcata nel caso in cui le mosse coinvolte all'interno dello stadio siano complesse e molteplici: in questo caso, un valore iniziale di α troppo piccolo potrebbe portare, all'inizio del processo di esplorazione, su soluzioni talmente inammissibili da risultare permanentemente compromesse, a causa dell'insufficiente numero di iterazioni miglioranti a disposizione.

Secondo Stadio

Detto ció, analizzando l'esplorazione compiuta dal secondo stadio risolutivo, si osserva che una buona percentuale dell'esplorazione é effettuata in regione non ammissibile. Si

tenta quindi di spingere l'esplorazione in regione ammissibile attraverso un aumento dell'alpha iniziale per il secondo stadio, attualmente a valore 1, tentando con valori compresi tra 1.25 e 4. Il miglior compromesso é stabilito a 2.0.

TABELLA 24. Risultati dopo il tuning del valore α iniziale (2.0) per il secondo stadio.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	1083	+616 ($\approx$ 132%)	199	99	22	$\approx$ 6
10P 30G D 4TW k1 BG30	506	1608	+1102 ($\approx$ 218%)	167	67	7	$\approx$ 4
10P 60G U 8TW k1 BG30	882	2095	+1213 ($\approx$ 138%)	227	127	51	$\approx$ 59
20P 30G S 4TW k1 BG15	285	562	+277 ($\approx$ 97%)	250	245	31	$\approx$ 27
20P 30G D 4TW k1 BG15	269	970	+701 ($\approx$ 261%)	192	92	16	$\approx$ 19
20P 60G U 8TW k1 BG30	454	1705	+1251 ($\approx$ 276%)	250	218	58	$\approx$ 57
30P 30G U 4TW k1 BG15	173	591	+418 ($\approx$ 242%)	250	227	38	$\approx$ 44
30P 30G S 8TW k1 BG60	195	1101	+906 ($\approx$ 465%)	250	249	41	$\approx$ 50
40P 60G S 8TW k1 BG60	262	1411	+1149 ($\approx$ 439%)	250	223	67	$\approx$ 71

Primo Stadio

Per quanto riguarda il primo stadio, si considera che le mosse impiegate sono molto piú reversibili, e quindi il rischio di compromettere l'esplorazione é minore. Vengono considerate quindi due diverse strategie di esplorazione:

(1) Nella prima strategia, *prudente*, visto il piccolo numero di passi a disposizione, si preferisce restare in territorio ammissibile; questo equivale ad aumentare il valore iniziale di α.
(2) Nella seconda strategia, *speculativa*, si predilige l'uscita dalla regione ammissibile nei passi iniziali, con la speranza di riuscire a rientrare prima che le condizioni di terminazione entrino in funzione; questo equivale a diminuire il valore iniziale di α.

TABELLA 26. Risultati per differenti valori iniziali di α per il primo stadio.

Problem	Improvement		
	$\alpha = 0.5$	$\alpha = 1.5$	$\alpha = 2.0$
10P 30G S 4TW k1 BG30	+490 ($\approx$ 105%)	+602 ($\approx$ **129%**)	+616 ($\approx$ **132%**)
10P 30G D 4TW k1 BG30	+1102 ($\approx$ 218%)	+1102 ($\approx$ 218%)	+1102 ($\approx$ 218%)
10P 60G U 8TW k1 BG30	+1001 ($\approx$ 113%)	+1226 ($\approx$ **139%**)	+1012 ($\approx$ 115%)
20P 30G S 4TW k1 BG15	+283 ($\approx$ 99%)	+286 ($\approx$ 100%)	+299 ($\approx$ **105%**)
20P 30G D 4TW k1 BG15	+707 ($\approx$ **263%**)	+701 ($\approx$ **261%**)	+666 ($\approx$ 248%)
20P 60G U 8TW k1 BG30	+1157 ($\approx$ 255%)	+1142 ($\approx$ 252%)	+1138 ($\approx$ 251%)
30P 30G U 4TW k1 BG15	+375 ($\approx$ 217%)	+390 ($\approx$ 225%)	+396 ($\approx$ **229%**)
30P 30G S 8TW k1 BG60	+906 ($\approx$ 465%)	+906 ($\approx$ 465%)	+906 ($\approx$ 465%)
40P 60G S 8TW k1 BG60	+1035 ($\approx$ 395%)	+1077 ($\approx$ **411%**)	+1075 ($\approx$ **410%**)

Con riferimento ai risultati precedenti, ottenuti con $\alpha = 1$, si decide di mantenersi leggermente in territorio ammissibile, tentando l'impostazione $\alpha = 1.10$.

Questa scelta determina miglioramenti sorprendenti, se si considera la vicinanza del valore ai tentativi 1.0 e 1.5. I risultati fanno decidere quindi un valore finale di 1.10. Riportiamo nella sezione successiva questi risultati.

1.6. Modalitá fast: parametri e risultati. Parametri e risultati relativi all'insieme campione per la modalitá fast (tempo di esecuzione $\approx 1'$).

TABELLA 28. Modalitá fast: parametri di terminazione.

Stadio	Parametro	Valore
Root-Level	Max steps	250
	Max steps no best	100
	Max rewind	4
Stadio 1 (Groups)	Max steps	50
	Max steps no best	15
Stadio 2 (Routes)	Max steps	190
	Max steps no best	35
Stadio 3 (Opt1)	Max steps	10
	Max steps no best	4

TABELLA 29. Modalitá fast: parametri relativi ai generatori di mosse.

Stadio	Generatore Impiegato	Parametro	Valore
Stadio 1 (Groups)	Group M.G.	Max Low-Profit Routes	6
		Max High-Load Routes	3
		Max Unsatisfied Groups	8
Stadio 2 (Routes)	FillPOI M.G.	Max Nearest POIs	16
		Max Highly-Profitable Routes	4
		Max Low-Timespan Routes	8
	SplitMerge M.G.	Max Split-Pool Routes	4
		Max Merge-Pool Routes	4
	SwapPOI M.G.	*none*	
Stadio 3 (Opt1)	Opt1 M.G.	*none*	

TABELLA 30. Modalitá fast: parametri di esplorazione.

Stadio	Parametro	Valore
Stadio 1 (Groups)	Initial α	1.10
	β	4
	θ	4
Stadio 2 (Routes)	Initial α	2.0
	β	5
	θ	8
Stadio 1 (Opt1)	Initial α	1.00
	β	1
	θ	1

TABELLA 31. Modalitá fast: risultati relativi all'insieme campione al termine del tuning.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	1069	+602 ($\approx$ 129%)	195	95	20	$\approx$ 5
10P 30G D 4TW k1 BG30	506	1608	+1102 ($\approx$ 218%)	167	67	8	$\approx$ 4
10P 60G U 8TW k1 BG30	882	2158	+1276 ($\approx$ 145%)	220	120	39	$\approx$ 54
20P 30G S 4TW k1 BG15	285	608	+323 ($\approx$ 113%)	219	119	32	$\approx$ 22
20P 30G D 4TW k1 BG15	269	970	+701 ($\approx$ 261%)	193	93	17	$\approx$ 19
20P 60G U 8TW k1 BG30	454	1689	+1235 ($\approx$ 272%)	250	212	68	$\approx$ 62
30P 30G U 4TW k1 BG15	173	577	+404 ($\approx$ 234%)	250	248	40	$\approx$ 42
30P 30G S 8TW k1 BG60	195	1101	+906 ($\approx$ 465%)	250	249	41	$\approx$ 55
40P 60G S 8TW k1 BG60	262	1407	+1145 ($\approx$ 437%)	250	211	67	$\approx$ 69

2. Modalitá accurate

Nella modalitá di funzionamento *accurate* (accurata) si richiede al risolutore di fornire soluzioni di ottima qualitá, senza imporre vincoli espliciti riguardo al tempo di esecuzione. Tuttavia, si desidera che esso rimanga entro valori accettabili, ovvero nell'ordine di qualche minuto.

Per imporre questo regime di funzionamento sará ovviamente necessario intervenire sulle condizioni di terminazione, che andranno rilassate. Sará inoltre opportuno espandere la dimensione del vicinato in modo tale da determinare, per ogni passo risolutivo, una ricerca piú ampia ed esaustiva: per far ció si agirá sui parametri relativi ai generatori di mosse.

2.1. Rilassamento delle condizioni di terminazione. Per aumentare il numero di passi di esplorazione innanzitutto é stato portato il limite di passi consentiti a 1000. In secondo luogo, occorre decidere se prolungare la durata dei singoli stadi oppure se aumentare il numero di rewind. Dato che si giudica soddisfacente e testata la dinamica di esplorazione raggiunta attraverso il tuning per la modalitá fast, la scelta ricade sulla seconda possibilitá. In questo modo, si mantiene ben saldo il principio di equilibrio su cui é fondato il risolutore: il principio che cerca di far progredire la risoluzione del problema di carico insieme a quello relativo al routing.

Per evitare per il secondo stadio lunghi prologhi dovuti alla condizione di terminazione sul numero massimo di passi senza miglioramenti, si é ridotto questo parametro.

Per quanto riguarda il terzo stadio, si é aumentato il numero di iterazioni consentite portandole al numero massimo di POI coinvolti nei problemi, in quanto questa modalitá di funzionamento deve garantire soluzioni buone anche dal punto di vista degli obiettivi secondari (timespan).

TABELLA 33. Modalitá fast: parametri di terminazione.

Stadio	Parametro	Valore
Root-Level	**Max steps**	~~250~~ **1000**
	Max steps no best	100
	Max rewind	~~4~~ **16**
Stadio 1 (Groups)	Max steps	50
	Max steps no best	15
Stadio 2 (Routes)	**Max steps**	~~190~~ **60**
	Max steps no best	35
Stadio 3 (Opt1)	**Max steps**	~~10~~ **40**
	Max steps no best	~~4~~ **10**

Si cerca un immediato riscontro attraverso una prova sull'insieme campione.

TABELLA 34. Risultati con condizioni di terminazione rilassate.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G S 4TW k1 BG30	467	1069	+602 (≈ 129%)	201	101	20	≈ 5
10P 30G D 4TW k1 BG30	506	1608	+1102 (≈ 218%)	167	67	8	≈ 4
10P 60G U 8TW k1 BG30	882	2158	+1276 (≈ 145%)	226	126	39	≈ 52
20P 30G S 4TW k1 BG15	285	608	+323 (≈ 113%)	222	122	32	≈ 19
20P 30G D 4TW k1 BG15	269	970	+701 (≈ 261%)	199	99	17	≈ 17
20P 60G U 8TW k1 BG30	454	**1715**	+1261 (≈ 278%)	382	282	72	≈ 88
30P 30G U 4TW k1 BG15	173	**600**	+427 (≈ 247%)	402	302	44	≈ 64
30P 30G S 8TW k1 BG60	195	**1146**	+951 (≈ 488%)	492	392	55	≈ 71
40P 60G S 8TW k1 BG60	262	1407	+1145 (≈ 437%)	295	195	67	≈ 76

Si nota come per quei problemi che raggiungevano al limite il massimo numero di iterazioni consentite siano giá presenti miglioramenti.

2.2. Espansione del vicinato. Per rendere l'esplorazione piú estensiva occorre aumentare le risorse fornite ai generatori di mosse. Tuttavia, si é verificato sperimentalmente che per i problemi di piccola dimensione e complessitá non si ottengono miglioramenti significativi.

Ci si sposta quindi su un insieme di benchmark esteso, nel quale si considerano problemi con numero di POI pari a 30 o 40, 60 gruppi di turisti, e Bus di capacitá 15 o 30, in modo tale da avere un grande numero di Route e POI coinvolti.

Per questi problemi si sono riscontrati miglioramenti con i seguenti parametri:

TABELLA 36. Espansione dell'esplorazione per problemi complessi.

Stadio	Generatore Impiegato	Parametro	Valore
Stadio 1 (Groups)	Group M.G.	Max Low-Profit Routes	6
		Max High-Load Routes	3
		Max Unsatisfied Groups	~~8~~ **20**
Stadio 2 (Routes)	FillPOI M.G.	**Max Nearest POIs**	~~16~~ **24**
		Max Highly-Profitable Routes	4
		Max Low-Timespan Routes	8
	SplitMerge M.G.	**Max Split-Pool Routes**	~~4~~ **12**
		Max Merge-Pool Routes	~~4~~ **12**
	SwapPOI M.G.	*none*	
Stadio 3 (Opt1)	Opt1 M.G.	*none*	

2.3. Modalitá accurate: parametri. Parametri relativi all'insieme campione per la modalitá accurate (tempo di esecuzione $> 1'$).

TABELLA 37. Modalitá accurate: parametri di terminazione.

Stadio	Parametro	Valore
Root-Level	Max steps	1000
	Max steps no best	100
	Max rewind	16
Stadio 1 (Groups)	Max steps	50
	Max steps no best	15
Stadio 2 (Routes)	Max steps	60
	Max steps no best	35
Stadio 3 (Opt1)	Max steps	40
	Max steps no best	10

TABELLA 38. Modalitá accurate: parametri relativi ai generatori di mosse.

Stadio	Generatore Impiegato	Parametro	Valore
Stadio 1 (Groups)	Group M.G.	Max Low-Profit Routes	6
		Max High-Load Routes	3
		Max Unsatisfied Groups	20
Stadio 2 (Routes)	FillPOI M.G.	Max Nearest POIs	24
		Max Highly-Profitable Routes	4
		Max Low-Timespan Routes	8
	SplitMerge M.G.	Max Split-Pool Routes	12
		Max Merge-Pool Routes	12
	SwapPOI M.G.	*none*	
Stadio 3 (Opt1)	Opt1 M.G.	*none*	

TABELLA 39. Modalitá accurate: parametri di esplorazione.

Stadio	Parametro	Valore
Stadio 1 (Groups)	Initial α	1.10
	β	4
	θ	4
Stadio 2 (Routes)	Initial α	2.0
	β	5
	θ	8
Stadio 1 (Opt1)	Initial α	1.00
	β	1
	θ	$+5$

CAPITOLO 10

Analisi dei risultati

Questa parte é dedicata all'analisi dei risultati ottenuti.

1. Confronto tra le modalitá

Complessivamente, la modalitá accurate ha portato un miglioramento statistico nella qualitá delle soluzioni, seppur marginale.

TABELLA 1. Variazione % del massimo miglioramento nelle due modalitá.

Class	Statistic	Mode	
		Fast	Accurate
All	μ	101.467	**101.469**
	σ	7.603	4.986
	Q_3 (75%)	100.000	**101.252**

Da ulteriori analisi emerge che il risolutore in modalitá accurate incontra particolari difficoltá in presenza di finestre temporali strette e preferenze non-dissimili tra i gruppi (quindi classi P e U). Considerando $k = 1$, questo equivale a dire che in questi problemi c'é in generale molta contesa tra i diversi gruppi per l'aggiudicarsi la visita dei POI "migliori", ma la presenza di finestre temporali ristrette impedisce la costruzione di Route lunghe ammissibili. Emerge il sospetto che il problema principale del risolutore in modalitá accurate sia da attribuire all'aumento eccessivo del numero di POI considerati dal generatore FillPOI, che effettua inserimenti in maniera troppo *greedy* nelle fasi iniziali di esecuzione.

Questo risultato rappresenta un'esperienza importante da tenere in considerazione negli eventuali sviluppi futuri di questo algoritmo, nonché di altri euristici: l'aumento della dimensione del vicinato é un miglioramento locale, che puó risultare talvolta controproducente in un'ottica globale.

A fronte dei risultati contrastanti forniti in modalitá accurate, si decide di adottare come set di parametri finali dell'algoritmo quelli determinati in modalitá fast. Nelle conclusioni 4 vengono indicati possibili approcci per aumentare la qualitá delle soluzioni combinando algoritmi di tipo fast.

Per l'elenco completo dei risultati finali su tutte le istanze di benchmark si rimanda al capitolo 11.

2. Statistiche

Sono state raccolte alcune statistiche riguardanti l'esecuzione dell'algoritmo sull'insieme di benchmark, sulla base di diverse classificazioni. In particolare:

- Nella prima colonna si riporta la classe considerata;
- La seconda colonna riguarda il rapporto fra il profitto della miglior soluzione con quello della soluzione iniziale.
- La terza colonna riguarda il profitto della soluzione iniziale.

- La quarta colonna riguarda i passi di esecuzione, e riporta il numero di passi totali compiuti e l'indice del passo per cui si é trovata la miglior soluzione.
- La quinta colonna riporta la percentuale di passi per cui si ha avuto un miglioramento della soluzione ottima corrente.

TABELLA 2. Distribuzione delle statistiche secondo diversi criteri di classificazione.

Class	p.Best / p.Initial		Initial Profit		Steps		#Best
	Mean	Max	Mean	Max	Total	Last Best	
All	(**403** $\pm$ 200)%	**1087**%	**355** $\pm$ 210	**991**	224 $\pm$ 33	148 $\pm$ 59	(**14** $\pm$ 6)%
10P	(298 $\pm$ 103)%	544%	**640** $\pm$ 220	**991**	213 $\pm$ 32	122 $\pm$ 47	(11 $\pm$ 4)%
20P	(361 $\pm$ 171)%	856%	370 $\pm$ 119	569	215 $\pm$ 37	131 $\pm$ 58	(14 $\pm$ 7)%
30P	(**478** $\pm$ 226)%	**1087**%	261 $\pm$ 93	386	235 $\pm$ 26	170 $\pm$ 58	(**16** $\pm$ 6)%
40P	(458 $\pm$ 218)%	1019%	197 $\pm$ 65	292	232 $\pm$ 32	164 $\pm$ 59	(15 $\pm$ 6)%
30G-D	(**557** $\pm$ 211)%	**995**%	252 $\pm$ 140	506	201 $\pm$ 35	114 $\pm$ 58	(9 $\pm$ 3)%
30G-S	(316 $\pm$ 103)%	565%	**280** $\pm$ 126	514	237 $\pm$ 20	170 $\pm$ 49	(**13** $\pm$ 3)%
30G-U	(410 $\pm$ 159)%	701%	248 $\pm$ 135	518	233 $\pm$ 23	151 $\pm$ 44	(12 $\pm$ 3)%
60G-D	(**508** $\pm$ 276)%	**1087**%	457 $\pm$ 237	958	221 $\pm$ 40	140 $\pm$ 59	(13 $\pm$ 4)%
60G-S	(273 $\pm$ 107)%	537%	**480** $\pm$ 247	**991**	229 $\pm$ 28	153 $\pm$ 59	(**20** $\pm$ 7)%
60G-U	(349 $\pm$ 148)%	680%	439 $\pm$ 208	882	225 $\pm$ 40	161 $\pm$ 72	(17 $\pm$ 7)%

Dalle statistiche emerge che l'ottimizzazione operata dall'algoritmo porta miglioramenti sostanziali ($\approx 4\times$ in media) alla soluzione iniziale, e che quindi risulta soddisfacente.

Dalle classificazioni effettuate si possono scorgere delle tendenze nei miglioramenti in base a certe caratteristiche del problema.

2.1. Classificazione per numero di POI. Innanzitutto, si osserva che i miglioramenti crescono all'aumentare del numero di POI, fino a raggiungere un valore massimo in corrispondenza della classe P30 (30 POI), per rimanere sostanzialmente costanti nella classe P40, probabilmente a causa dell'insorgenza di fenomeni di saturazione introdotti dalle finestre temporali.

Si nota inoltre che, al contrario, il generatore di soluzione é caratterizzato da un costante peggioramento all'aumentare dei POI, in maniera sostanzialmente inversamente proporzionale. Questo é dovuto all'approccio costruttivo impiegato per la generazione delle istanze: i profitti associati ad ogni gruppo di turisti sono infatti normalizzati per avere una somma complessiva pari a 100. É quindi naturale che il generatore di soluzione iniziale, associando un POI ad ogni Route, determini - nel caso in cui effettivamente operi in maniera uniforme - profitti inversamente proporzionali al numero di POI presenti. Questa tendenza quindi offre un primo riscontro sull'uniformitá con cui il generatore di soluzione iniziale opera.

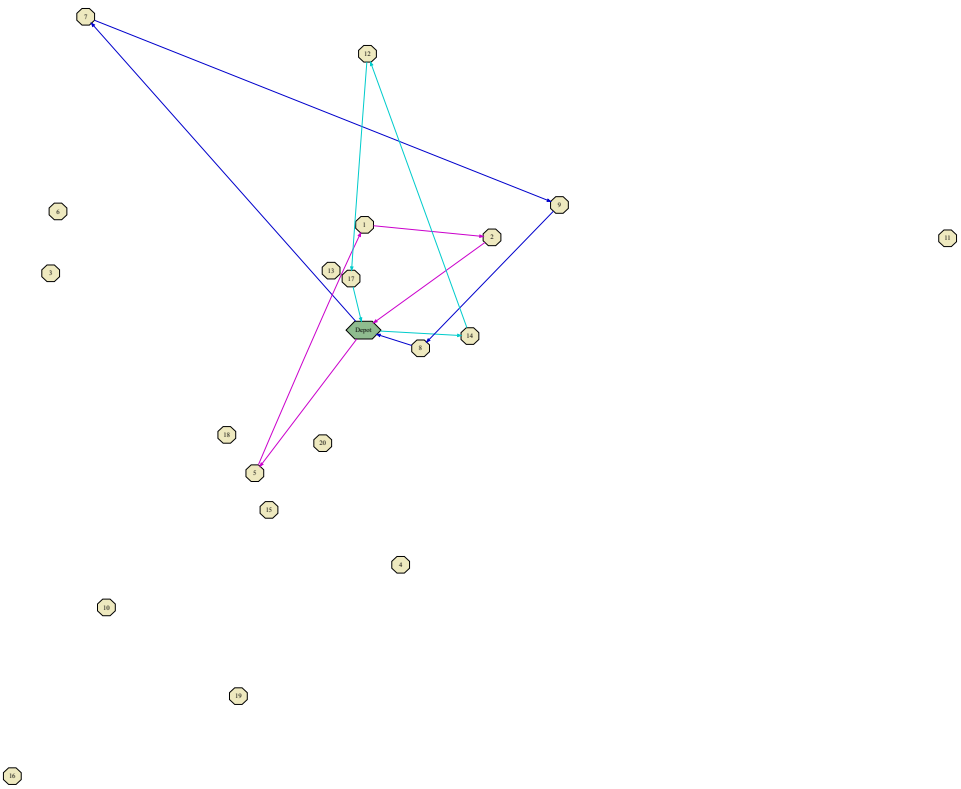
2.2. Classificazione per numero di gruppi di turisti - Classe di profitti. In questa classificazione emerge che i miglioramenti maggiori si hanno per la classe di profitti D (Dissimilar), seguita dalla classe U (Uniform), indipendentemente dal numero di gruppi

di turisti. Questo é spiegabile considerando che la contesa dei POI da parte dei gruppi, particolarmente influente nel caso $k = 1$ (ovvero visite esclusive), é piú problematica da soddisfare quando i gruppi condividono le stesse preferenze, o quando queste sono distribuite uniformemente. Nel caso di preferenze dissimili, é piú semplice partizionare i gruppi in Route diverse, e far visitare a queste solo i POI piú profittevoli.

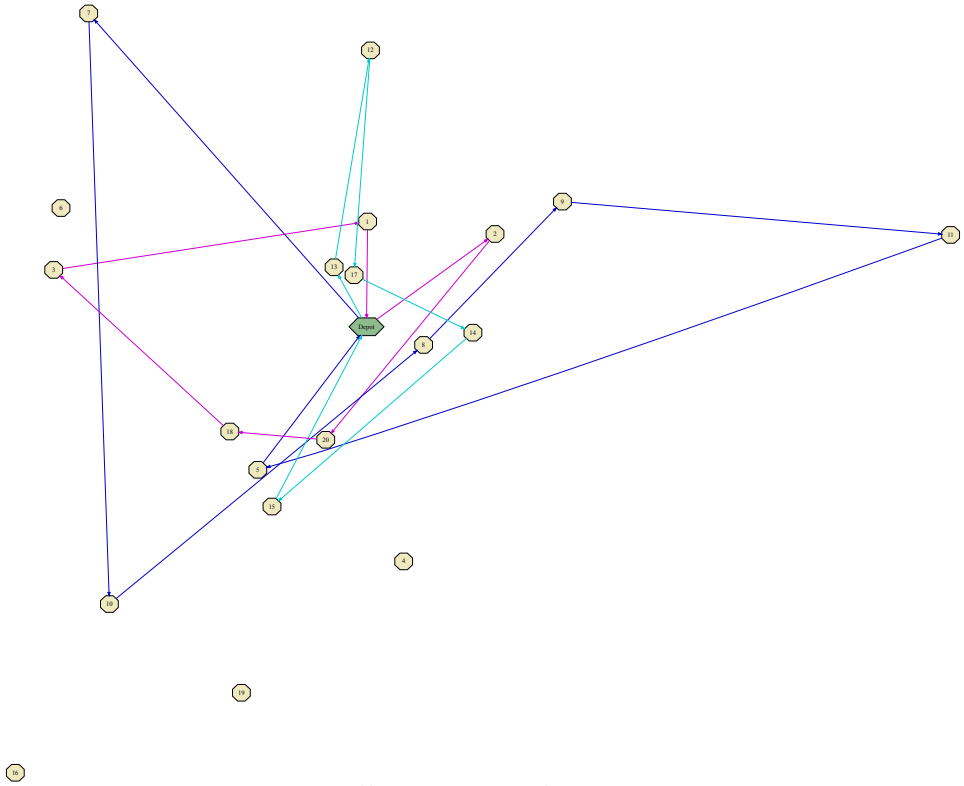
Si osserva inoltre che il generatore di soluzione iniziale ha la massima efficienza per la classe S di profitti. Questo é dovuto al fatto che l'algoritmo di bin-packing impiegato per l'assegnazione iniziale dei gruppi ai Bus opera solo sulla base del carico, e non dei profitti: é quindi naturale che profitti simili diano origine a carichi iniziali con gruppi statisticamente piú compatibili.

3. Esempio di soluzione

Riportiamo qui due soluzioni per due istanze di problemi che differiscono solo per la finestra temporale considerata (4TW - 8TW). Il motivo di questa scelta é che la variante con finestre temporali brevi é caratterizzata da una percentuale di miglioramento nella media (297%); la variante 8TW é stata introdotta per rendere l'idea di come la soluzione si modifichi al variare delle finestre temporali.



Problem_BTP20P-30G-D-4TW-k1-BG30_30



Problem_BTP20P-30G-D-8TW-k1-BG30_30

TABELLA 4. Miglior soluzione per l'istanza BTP20P-30G-D-4TW-k1-BG30_30.

Route	POI	Arrival	Departure	Profit
#0	#0	0	0	0
	#14	7 / 180	82	128
	#12	104 / 240	164	98
	#17	181 / 240	226	91
	#0	**230 / 240**	//	0
	Total Profit			**317**
	Groups			10
	Load			**30 / 30**
#1	#0	0	0	0
	#7	29 / 120	89	129
	#9	123 / 240	183	120
	#8	197 / 240	227	128
	#0	**230 / 240**	//	0
	Total Profit			**377**
	Groups			10
	Load			**30 / 30**
#2	#0	0	0	0
	#5	13 / 120	73	59
	#1	93 / 240	183	132
	#2	191 / 240	206	123
	#0	**216 / 240**	//	0
	Total Profit			**314**
	Groups			10
	Load			**30 / 30**

TABELLA 6. Miglior soluzione per l'istanza BTP20P-30G-D-8TW-k1-BG30_30.

Route	POI	Arrival	Departure	Profit
#0	#0	0	0	0
	#13	4 / 180	109	120
	#12	125 / 420	185	98
	#17	202 / 480	247	91
	#14	255 / 300	330	128
	#15	348 / 480	438	129
	#0	**453 / 480**	//	0
	Total Profit			**566**
	Groups			10
	Load			**30 / 30**
#1	#0	0	0	0
	#7	29 / 300	89	129
	#10	133 / 180	148	98
	#8	176 / 240	206	128
	#9	220 / 240	280	120
	#11	305 / 480	350	75
	#5	398 / 480	458	91
	#0	**471 / 480**	//	0
	Total Profit			**641**
	Groups			10
	Load			**30 / 30**
#2	#0	0	0	0
	#2	10 / 180	75	123
	#20	94 / 480	199	123
	#18	205 / 360	250	83
	#3	266 / 420	356	106
	#1	376 / 480	466	132
	#0	**474 / 480**	//	0
	Total Profit			**567**
	Groups			10
	Load			**30 / 30**

Parte 4

Conclusioni

I risultati sperimentali hanno messo in evidenza la validitá di principio dell'algoritmo nell'insieme di problemi di benchmark.

Si vogliono indicare alcune direzioni di sviluppo future per quanto riguarda il BTP, l'algoritmo qui presentato, ed il framework sperimentale che ha reso possibile la fase sperimentale ed implementativa.

Sviluppi futuri

Framework

Il framework é stato sviluppato per rispondere alle continue esigenze di riscontri riguardo le ipotesi risolutive; a causa di ció, esso é in uno stadio di maturitá embrionale. Sebbene sia stato sviluppato per la maggior parte in maniera attenta, in alcuni casi, per le risorse a disposizione, é stato scelto di dedicare piú tempo allo sviluppo dell'algoritmo che del framework. In sostanza, riguardo esso, si vedono due grandi alternative: esso puó essere rifattorizzato ed ampliato, oppure si puó giudicare come uno strumento sperimentale che, come tale, ha compiuto il proprio obiettivo, e quindi cercare un framework esistente maturo su cui sia possibile effettuare il porting dell'algoritmo risolutivo, e successivamente continuare qui lo sviluppo dell'algoritmo.

Algoritmo

Per quanto riguarda lo sviluppo dell'algoritmo, ci sono alcuni punti che vale la pena menzionare. Innanzitutto, il fatto che si disponga di un tuning particolarmente efficiente per risolvere problemi in tempo breve non preclude necessariamente la possibilitá di trovare soluzioni migliori. Un algoritmo di breve durata, infatti, ha il pregio di poter essere impiegato come stadio risolutivo in un approccio *multi-start*[1], quando sia necessario focalizzarsi sulla produzione di soluzioni di altissima qualitá: in questo setting, disponendo di una parametrizzazione buona per la generazione di una soluzione iniziale, possono venir lanciati simultaneamente molte istanze dell'algoritmo ciascuna delle quali parte da un punto differente dello spazio delle soluzioni. Questo approccio é spesso impiegato come moltiplicatore dell'efficacia di un algoritmo semplice.

I principali raffinamenti che potrebbero migliorare sostanzialmente l'algoritmo consistono nel concetto di *elite solutions*, e in particolare la tecnica del *path relinking*, proposta inizialmente da Glover[2]. Queste tecniche porterebbero una componente di memoria a lungo termine al processo di risoluzione, che si pensa sia la componente al momento piú carente.

Generatori

Per quanto riguarda la generazione di mosse, si é giá messo in evidenza come le macromosse di split e merge siano quelle piú critiche per il raggiungimento di soluzioni di buona qualitá. Vale la pena quindi di tentare ad esempio un raffinamento dell'euristica contenuta nella mossa di merge. Quello che manca, inoltre, é un meccanismo per equalizzare i punteggi di mosse di tipo diverso: per quanto il calcolo dell'indice di non-ammissibilitá possa essere assente da bias, occorre una politica di arbitraggio che garantisca una certa *fairness* tra tutti i tipi di mosse generate.

Riguardo al Bus Touring Problem, é stato messo in evidenza come questo generalizzi altri problemi di ottimizzazione. Sarebbe quindi molto proficuo valutare le performance degli algoritmi risolutivi sviluppati per il BTP su tali problemi. Questo richiederebbe in sostanza di tradurre un set ben noto di istanze di benchmark per un problema specifico in formulazione BTP. Quello che ci si attende é che gli algoritmi pensati per i problemi specifici siano in linea di massima piú efficienti; resta, comunque, l'interrogativo sull'entitá dello scarto.

---

[1] Rafael Marti. *Handbook of Metaheuristics*. 2003.

[2] Fred W. Glover. «Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges». In: *Interfaces in Computer Science and Operations Research*. 1996, pp. 1–75.

Nota dell'autore

L'esperienza nella progettazione di un algoritmo euristico é un lungo percorso sperimentale, nel quale dall'analisi del problema emergono osservazioni, ipotesi, congetture, ed é proprio il continuo tentativo di verificarle o smentirle che segna in ogni istante la direzione di questo percorso.

Nel corso di questa esperienza ci si é imbattuti spesso in concetti ed argomenti che per l'interesse suscitato ne avrebbero certamente meritato l'esplorazione. Tuttavia, questo avrebbe comportato una serie di escursioni che, per quanto affascinanti ed istruttive, avrebbero sicuramente minato il progresso riguardo gli obiettivi prefissi.

Pertanto, é seguendo la stessa filosofia su cui si basa il nostro algoritmo che si é cercato sempre di tornare, ad un certo punto, nella regione ammissibile.

Parte 5

Appendice

CAPITOLO 11

Risultati estesi

Seguono le soluzioni dell'intero problem-set di benchmark trovate dal risolutore impostato in modalitá fast.

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
10P 30G U 4TW k1 BG15	518	986	+468 (≈ 90%)	250	166	17	≈ 18
10P 30G S 4TW k1 BG15	445	663	+218 (≈ 49%)	250	176	21	≈ 23
10P 30G D 4TW k1 BG15	490	1215	+725 (≈ 148%)	181	81	11	≈ 13
10P 30G U 8TW k1 BG15	518	879	+361 (≈ 70%)	250	161	18	≈ 10
10P 30G S 8TW k1 BG15	445	811	+366 (≈ 82%)	250	175	22	≈ 10
10P 30G D 8TW k1 BG15	490	1402	+912 (≈ 186%)	183	83	10	≈ 13
10P 30G U 4TW k1 BG30	437	1224	+787 (≈ 180%)	250	189	25	≈ 11
10P 30G S 4TW k1 BG30	467	1069	+602 (≈ 129%)	195	95	20	≈ 7
10P 30G D 4TW k1 BG30	506	1608	+1102 (≈ 218%)	167	67	8	≈ 6
10P 30G U 8TW k1 BG30	437	1531	+1094 (≈ 250%)	250	177	21	≈ 12
10P 30G S 8TW k1 BG30	467	1240	+773 (≈ 166%)	220	120	25	≈ 8
10P 30G D 8TW k1 BG30	506	2188	+1682 (≈ 332%)	168	68	14	≈ 6
10P 60G U 4TW k1 BG30	882	1968	+1086 (≈ 123%)	250	201	41	≈ 79
10P 60G S 4TW k1 BG30	937	1631	+694 (≈ 74%)	231	131	44	≈ 49
10P 60G D 4TW k1 BG30	958	2268	+1310 (≈ 137%)	152	52	21	≈ 38
10P 60G U 8TW k1 BG30	882	2158	+1276 (≈ 145%)	220	120	39	≈ 55
10P 60G S 8TW k1 BG30	937	1849	+912 (≈ 97%)	210	110	35	≈ 30
10P 60G D 8TW k1 BG30	958	2900	+1942 (≈ 203%)	250	178	25	≈ 64
10P 30G U 4TW k1 BG60	425	1387	+962 (≈ 226%)	176	76	17	≈ 4
10P 30G S 4TW k1 BG60	514	1342	+828 (≈ 161%)	196	96	24	≈ 6
10P 30G D	420	1536	+1116	167	67	12	≈ 5

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
4TW k1 BG60			(≈ 266%)				
10P 30G U 8TW k1 BG60	425	1975	+1550 (≈ 365%)	213	113	25	≈ 7
10P 30G S 8TW k1 BG60	514	1769	+1255 (≈ 244%)	198	98	26	≈ 6
10P 30G D 8TW k1 BG60	420	2286	+1866 (≈ 444%)	171	71	17	≈ 5
10P 60G U 4TW k1 BG60	770	2522	+1752 (≈ 228%)	212	112	29	≈ 12
10P 60G S 4TW k1 BG60	991	2271	+1280 (≈ 129%)	211	111	40	≈ 12
10P 60G D 4TW k1 BG60	833	3346	+2513 (≈ 302%)	250	155	24	≈ 14
10P 60G U 8TW k1 BG60	770	3122	+2352 (≈ 305%)	250	232	46	≈ 17
10P 60G S 8TW k1 BG60	991	2695	+1704 (≈ 172%)	212	112	39	≈ 11
10P 60G D 8TW k1 BG60	833	4403	+3570 (≈ 429%)	194	94	22	≈ 13
20P 30G U 4TW k1 BG15	269	656	+387 (≈ 144%)	197	97	20	≈ 22
20P 30G S 4TW k1 BG15	285	608	+323 (≈ 113%)	219	119	32	≈ 23
20P 30G D 4TW k1 BG15	269	970	+701 (≈ 261%)	193	93	17	≈ 23
20P 30G U 8TW k1 BG15	269	853	+584 (≈ 217%)	212	112	22	≈ 22
20P 30G S 8TW k1 BG15	285	666	+381 (≈ 134%)	235	135	24	≈ 21
20P 30G D 8TW k1 BG15	269	1100	+831 (≈ 309%)	250	161	21	≈ 21
20P 60G U 4TW k1 BG15	533	782	+249 (≈ 47%)	132	32	8	≈ 9
20P 60G S 4TW k1 BG15	452	504	+52 (≈ 12%)	135	35	11	≈ 13
20P 60G D 4TW k1 BG15	569	1107	+538 (≈ 95%)	234	134	14	≈ 24
20P 60G U 8TW k1 BG15	533	577	+44 (≈ 8%)	114	14	5	≈ 8
20P 60G S 8TW k1 BG15	452	759	+307 (≈ 68%)	191	91	32	≈ 13

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
20P 60G D 8TW k1 BG15	569	1108	+539 (≈ 95%)	171	71	18	≈ 15
20P 30G U 4TW k1 BG30	235	785	+550 (≈ 234%)	229	129	23	≈ 21
20P 30G S 4TW k1 BG30	295	799	+504 (≈ 171%)	250	190	27	≈ 23
20P 30G D 4TW k1 BG30	254	1008	+754 (≈ 297%)	206	106	10	≈ 18
20P 30G U 8TW k1 BG30	235	1208	+973 (≈ 414%)	226	126	25	≈ 16
20P 30G S 8TW k1 BG30	295	1110	+815 (≈ 276%)	250	222	47	≈ 24
20P 30G D 8TW k1 BG30	254	1774	+1520 (≈ 598%)	178	78	20	≈ 16
20P 60G U 4TW k1 BG30	454	1239	+785 (≈ 173%)	247	147	56	≈ 76
20P 60G S 4TW k1 BG30	484	1007	+523 (≈ 108%)	224	124	61	≈ 68
20P 60G D 4TW k1 BG30	495	1959	+1464 (≈ 296%)	250	225	34	≈ 77
20P 60G U 8TW k1 BG30	454	1689	+1235 (≈ 272%)	250	212	68	≈ 74
20P 60G S 8TW k1 BG30	484	1362	+878 (≈ 181%)	250	249	79	≈ 74
20P 60G D 8TW k1 BG30	495	2656	+2161 (≈ 437%)	250	161	43	≈ 62
20P 30G U 4TW k1 BG60	222	729	+507 (≈ 228%)	250	160	24	≈ 16
20P 30G S 4TW k1 BG60	293	897	+604 (≈ 206%)	250	164	29	≈ 13
20P 30G D 4TW k1 BG60	211	848	+637 (≈ 302%)	172	72	14	≈ 12
20P 30G U 8TW k1 BG60	222	1255	+1033 (≈ 465%)	197	97	24	≈ 20
20P 30G S 8TW k1 BG60	293	1357	+1064 (≈ 363%)	205	105	26	≈ 19
20P 30G D 8TW k1 BG60	211	1464	+1253 (≈ 594%)	186	86	19	≈ 18
20P 60G U 4TW k1 BG60	413	1397	+984 (≈ 238%)	250	151	36	≈ 34
20P 60G S	500	1174	+674	220	120	41	≈ 29

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
4TW k1 BG60			(≈ 135%)				
20P 60G D 4TW k1 BG60	422	2176	+1754 (≈ 416%)	192	92	18	≈ 24
20P 60G U 8TW k1 BG60	413	2260	+1847 (≈ 447%)	250	211	57	≈ 35
20P 60G S 8TW k1 BG60	500	1932	+1432 (≈ 286%)	250	150	56	≈ 30
20P 60G D 8TW k1 BG60	422	3614	+3192 (≈ 756%)	250	237	41	≈ 27
30P 30G U 4TW k1 BG15	173	577	+404 (≈ 234%)	250	248	40	≈ 46
30P 30G S 4TW k1 BG15	197	516	+319 (≈ 162%)	250	177	30	≈ 53
30P 30G D 4TW k1 BG15	181	857	+676 (≈ 373%)	203	103	15	≈ 37
30P 30G U 8TW k1 BG15	173	805	+632 (≈ 365%)	250	183	33	≈ 37
30P 30G S 8TW k1 BG15	197	663	+466 (≈ 237%)	250	184	42	≈ 29
30P 30G D 8TW k1 BG15	181	1241	+1060 (≈ 586%)	250	241	30	≈ 29
30P 60G U 4TW k1 BG15	386	808	+422 (≈ 109%)	250	232	18	≈ 44
30P 60G S 4TW k1 BG15	356	697	+341 (≈ 96%)	250	159	26	≈ 35
30P 60G D 4TW k1 BG15	378	575	+197 (≈ 52%)	155	55	6	≈ 27
30P 60G U 8TW k1 BG15	386	883	+497 (≈ 129%)	190	90	29	≈ 16
30P 60G S 8TW k1 BG15	356	639	+283 (≈ 79%)	238	138	30	≈ 30
30P 60G D 8TW k1 BG15	378	1039	+661 (≈ 175%)	212	112	29	≈ 28
30P 30G U 4TW k1 BG30	159	652	+493 (≈ 310%)	250	162	28	≈ 32
30P 30G S 4TW k1 BG30	200	630	+430 (≈ 215%)	227	127	31	≈ 28
30P 30G D 4TW k1 BG30	172	999	+827 (≈ 481%)	250	245	48	≈ 34
30P 30G U 8TW k1 BG30	159	1030	+871 (≈ 548%)	250	159	35	≈ 37

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
30P 30G S 8TW k1 BG30	200	942	+742 (≈ 371%)	250	213	44	≈ 33
30P 30G D 8TW k1 BG30	172	1454	+1282 (≈ 745%)	186	86	19	≈ 27
30P 60G U 4TW k1 BG30	356	1221	+865 (≈ 243%)	250	250	69	≈ 81
30P 60G S 4TW k1 BG30	378	980	+602 (≈ 159%)	237	137	59	≈ 85
30P 60G D 4TW k1 BG30	330	1646	+1316 (≈ 399%)	250	215	41	≈ 88
30P 60G U 8TW k1 BG30	356	1624	+1268 (≈ 356%)	250	159	57	≈ 75
30P 60G S 8TW k1 BG30	378	1320	+942 (≈ 249%)	250	249	87	≈ 73
30P 60G D 8TW k1 BG30	330	2676	+2346 (≈ 711%)	220	120	46	≈ 54
30P 30G U 4TW k1 BG60	152	694	+542 (≈ 357%)	247	147	24	≈ 28
30P 30G S 4TW k1 BG60	195	711	+516 (≈ 265%)	250	203	32	≈ 24
30P 30G D 4TW k1 BG60	142	880	+738 (≈ 520%)	185	85	21	≈ 19
30P 30G U 8TW k1 BG60	152	1064	+912 (≈ 600%)	250	159	32	≈ 43
30P 30G S 8TW k1 BG60	195	1101	+906 (≈ 465%)	250	249	41	≈ 45
30P 30G D 8TW k1 BG60	142	1384	+1242 (≈ 875%)	183	83	26	≈ 29
30P 60G U 4TW k1 BG60	279	1322	+1043 (≈ 374%)	250	222	55	≈ 42
30P 60G S 4TW k1 BG60	381	1280	+899 (≈ 236%)	212	112	45	≈ 42
30P 60G D 4TW k1 BG60	283	1925	+1642 (≈ 580%)	250	156	24	≈ 42
30P 60G U 8TW k1 BG60	279	1897	+1618 (≈ 580%)	250	234	60	≈ 48
30P 60G S 8TW k1 BG60	381	1908	+1527 (≈ 401%)	250	248	65	≈ 44
30P 60G D 8TW k1 BG60	283	3075	+2792 (≈ 987%)	250	185	37	≈ 47
40P 30G U	147	378	+231	250	225	32	≈ 48

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
4TW k1 BG15			(≈ 157%)				
40P 30G S 4TW k1 BG15	156	326	+170 (≈ 109%)	250	247	26	≈ 49
40P 30G D 4TW k1 BG15	139	589	+450 (≈ 324%)	250	188	25	≈ 53
40P 30G U 8TW k1 BG15	147	595	+448 (≈ 305%)	210	110	32	≈ 55
40P 30G S 8TW k1 BG15	156	532	+376 (≈ 241%)	250	227	42	≈ 63
40P 30G D 8TW k1 BG15	139	940	+801 (≈ 576%)	250	162	26	≈ 60
40P 60G U 4TW k1 BG15	292	574	+282 (≈ 97%)	177	77	11	≈ 55
40P 60G S 4TW k1 BG15	271	520	+249 (≈ 92%)	213	113	25	≈ 62
40P 60G D 4TW k1 BG15	288	456	+168 (≈ 58%)	121	21	7	≈ 33
40P 60G U 8TW k1 BG15	292	887	+595 (≈ 204%)	187	87	19	≈ 22
40P 60G S 8TW k1 BG15	271	647	+376 (≈ 139%)	250	153	31	≈ 34
40P 60G D 8TW k1 BG15	288	1144	+856 (≈ 297%)	250	195	31	≈ 35
40P 30G U 4TW k1 BG30	120	467	+347 (≈ 289%)	228	128	30	≈ 32
40P 30G S 4TW k1 BG30	157	415	+258 (≈ 164%)	250	200	28	≈ 24
40P 30G D 4TW k1 BG30	129	632	+503 (≈ 390%)	137	37	12	≈ 14
40P 30G U 8TW k1 BG30	120	698	+578 (≈ 482%)	250	215	39	≈ 50
40P 30G S 8TW k1 BG30	157	688	+531 (≈ 338%)	250	229	42	≈ 43
40P 30G D 8TW k1 BG30	129	1081	+952 (≈ 738%)	250	157	24	≈ 48
40P 60G U 4TW k1 BG30	241	791	+550 (≈ 228%)	240	140	57	≈ 99
40P 60G S 4TW k1 BG30	269	689	+420 (≈ 156%)	250	214	55	≈ 99
40P 60G D 4TW k1 BG30	254	1239	+985 (≈ 388%)	250	179	40	≈ 99

Problem	Profits			Steps		N. of Best	Time (s)
	Initial	Best	Improvement	Total	Best at		
40P 60G U 8TW k1 BG30	241	1185	+944 (≈ 392%)	250	243	71	≈ 115
40P 60G S 8TW k1 BG30	269	1005	+736 (≈ 274%)	250	249	82	≈ 111
40P 60G D 8TW k1 BG30	254	2029	+1775 (≈ 699%)	237	137	45	≈ 110
40P 30G U 4TW k1 BG60	115	479	+364 (≈ 317%)	195	95	24	≈ 18
40P 30G S 4TW k1 BG60	161	437	+276 (≈ 171%)	243	143	22	≈ 19
40P 30G D 4TW k1 BG60	106	649	+543 (≈ 512%)	250	207	20	≈ 21
40P 30G U 8TW k1 BG60	115	806	+691 (≈ 601%)	250	185	36	≈ 43
40P 30G S 8TW k1 BG60	161	721	+560 (≈ 348%)	250	197	33	≈ 43
40P 30G D 8TW k1 BG60	106	1055	+949 (≈ 895%)	208	108	27	≈ 40
40P 60G U 4TW k1 BG60	228	846	+618 (≈ 271%)	222	122	24	≈ 36
40P 60G S 4TW k1 BG60	262	870	+608 (≈ 232%)	250	153	40	≈ 36
40P 60G D 4TW k1 BG60	213	1368	+1155 (≈ 542%)	225	125	32	≈ 34
40P 60G U 8TW k1 BG60	228	1381	+1153 (≈ 506%)	250	243	50	≈ 65
40P 60G S 8TW k1 BG60	262	1407	+1145 (≈ 437%)	250	211	67	≈ 64
40P 60G D 8TW k1 BG60	213	2171	+1958 (≈ 919%)	250	180	38	≈ 69

Bibliografia

[Bia+09] L. Bianchi et al. «A survey on metaheuristics for stochastic combinatorial optimization». In: *Natural Computing: an international journal* (2009), pp. 238–287.

[CGW96a] I. Chao, B. Golden e E. Wasil. «Theory and metodology - a fast and effective heuristic for the orienteering problem.» In: *European Journal of Operational Research* 88 (1996), pp. 475, 489.

[CGW96b] I. Chao, B. Golden e E. Wasil. «Theory and metodology - the team orienteering problem.» In: *European Journal of Operational Research* 88 (1996), pp. 464, 474.

[Glo86] Fred W. Glover. «Future Paths for Integer Programming and Links to Artificial Intelligence». In: *Computers and Operations Research* 13 (1986), pp. 533–549.

[Glo89] Fred W. Glover. «Tabu Search - Part I». In: *ORSA Journal on Computing* 1 (1989), pp. 190–216.

[Glo96] Fred W. Glover. «Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges». In: *Interfaces in Computer Science and Operations Research*. 1996, pp. 1–75.

[Kor02] Richard E. Korf. «A New Algorithm for Optimal Bin Packing». In: *AAAI Proceedings*. 2002.

[Mar03] Rafael Marti. *Handbook of Metaheuristics*. 2003.

[MT90] Silvano Martello e Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations (Wiley Series in Discrete Mathematics and Optimization)*. 1990.

[RB03] A. Roli e C. Blum. «Metaheuristics in combinatorial optimization: Overview and conceptual comparison». In: *ACM Computing Surveys* (2003), pp. 268–308.

[VSVO11] Pieter Vansteenwegen, Wouter Souffriau e Dirk Van Oudheusden. «The orienteering problem: A survey». In: *European Journal of Operational Research* 209 (2011), pp. 1–10.