

UNIVERSITA' DEGLI STUDI DI BRESCIA
CORSO DI LAUREA MAGISTRALE



ALGORITMI DI OTTIMIZZAZIONE

The Clustered Team Orienteering Problem
With Services Sequences

GRUPPO:

Angela Beltramelli (704849)

Davide Lonati (705990)

Caterina Pezzaioli (705405)

Francesco Piazza (77205)

Francesca Varisco (706104)

Giulia Zanoni (706021)

ANNO ACCADEMICO 2017-2018

SOMMARIO

1. Problema e Modellizzazione Matematica	1
2. Soluzione proposta	2
3. Miglioramento della modellizzazione.....	3
4. Algoritmo Costruttivo	5
5. Metaeuristica	6
5.1 Principali Parametri	6
5.2 Euristiche di Distruzione	7
5.3 Euristiche di Riparazione	7
5.4 Assegnazione e Aggiornamento dei Pesi	8
6. Cluster Roulette	9
7. Simulated Annealing	10
8. Local Search	11
9. Taratura dei Parametri.....	11
9.1 Initial q	12
9.2 Delta q.....	13
9.3 Max Iterations w/o improvement in a segment.....	13
9.4 Lambda	14
9.5 Alpha	15
9.6 Cooldown Gamma	15
9.7 Warmup Gamma	16
9.8 Risultati complessivi.....	17
10. Analisi.....	19
Bibliografia	22

1. PROBLEMA E MODELLIZZAZIONE MATEMATICA

In questa sezione viene esposto il problema in esame nelle sue principali caratteristiche.

L'*Orienteering Problem* [1] (OP) prende il nome da uno sport praticato all'aperto: dato un insieme di punti di controllo, con associato ciascuno un punteggio, i concorrenti devono visitare un sottoinsieme di questi in modo da massimizzare il punteggio totale e da arrivare al punto di arrivo entro un certo periodo di tempo predefinito.

Il problema in esame è un *Team Orienteering Problem* (TOP) dove, a differenza dell'OP, non vi sono singoli concorrenti ma squadre composte da più persone/veicoli. È comparso per la prima volta in letteratura con Butt e Cavalier [2] (1994) con il nome di *Multiple Tour Maximum Collection Problem* e due anni dopo è stato formalmente presentato [3]. Rispetto al TOP, i punti di controllo sono costituiti da più nodi che sono raggruppati insieme tramite cluster. Il punteggio non è più associato al singolo nodo, ma ad un cluster.

La modellizzazione iniziale prevedeva la seguente funzione obiettivo ed i seguenti vincoli:

$$\max \sum_{c \in C} p_c y_c \quad (1)$$

$$\sum_{(0,i) \in \delta^+(0)} x_{ij}^v = \sum_{(i,n_{max}+1) \in \delta^-(n_{max}+1)} x_{i,n_{max}+1}^v = 1 \quad \forall v \in V \quad (2)$$

$$\sum_{v \in V} \sum_{(i,j) \in \delta^+(i)} x_{ij}^v = \sum_{v \in V} \sum_{(j,i) \in \delta^-(i)} x_{ji}^v = y_c \quad \forall c \in \{1, \dots, c_{max}\}, i \in C_c \quad (3)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^v = \sum_{(j,i) \in \delta^-(i)} x_{ji}^v \quad \forall v \in V, i \in N \quad (4)$$

$$\sum_{(i,j) \in \delta^+(i)} z_{ij} - \sum_{(j,i) \in \delta^-(i)} z_{ji} = \sum_{v \in V} \sum_{(i,j) \in \delta^+(i)} (t_{ij} + d_i) x_{ij}^v \quad \forall i \in N \quad (5)$$

$$z_{0i} = t_{0i} \sum_{v \in V} x_{0i}^v \quad \forall (0,i) \in \delta^+(0) \quad (6)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^v + \sum_{(j,i) \in \delta^-(i)} x_{ji}^v \leq 2a_{iv} \quad \forall v \in V, i \in N \quad (7)$$

$$z_{ij} \leq t_{max} \sum_{v \in V} x_{ij}^v \quad \forall i \in N^+, (i,j) \in \delta^+(i) \quad (8)$$

$$w_{ij} \left(\sum_{(k,i) \in \delta^-(i)} z_{ki} + d_i \sum_{v \in V} \sum_{(k,i) \in \delta^-(i)} x_{ki}^v \right) \leq \sum_{(k,j) \in \delta^-(j)} z_{kj} \quad \forall c \in \{1, \dots, c_{max}\}, i, j \in C_c \quad (9)$$

$$z_{ij} \geq 0 \quad \forall i \in N^+, (i,j) \in \delta^+(i) \quad (10)$$

$$x_{ij}^k \in \{0,1\} \quad \forall i \in N^+, (i,j) \in \delta^+(i), v \in V \quad (11)$$

$$y_c \in \{0,1\} \quad \forall c \in \{1, \dots, c_{max}\}, \quad (12)$$

2. SOLUZIONE PROPOSTA

Dopo aver analizzato il problema, si è deciso di fissare come punto di partenza della soluzione un classico algoritmo di *Adaptive Large Neighborhood Search* [4] (ALNS). Tale algoritmo è una metaeuristica iterativa basata sull'applicazione, ad ogni iterazione, di un'euristica di distruzione e di una di ricostruzione, con l'obiettivo di cercare soluzioni migliori rispetto a quella corrente. La scelta di utilizzare questo metodo è stata presa dopo aver esaminato attentamente il problema, individuandone caratteristiche e particolarità. Innanzitutto, la ALNS è stata sviluppata per la prima volta da Ropke e Pisinger su un problema di *Pickup and Delivery* con finestra di tempo limitata [4], ottenendo nel 50% dei casi miglioramenti sui benchmark di 350 istanze. Anche nel nostro problema, la risorsa scarsa è il tempo, che deve essere ottimizzato al fine di massimizzare il punteggio totale. Osservando, inoltre, la letteratura a nostra disposizione, ci si è accorti come nei problemi di *Team Orienteering* si preferisse un approccio più classico basato su [Tabu Search](#) ed euristiche di ricerca locale, trascurando modelli più recenti come la ALNS (1996) a causa, anche, della sua difficoltà implementativa. Infine, si è prediletto l'utilizzo della ALNS in quanto, come analizzato più avanti, garantisce una maggior flessibilità rispetto ad una *Large Neighborhood Search* (LNS), algoritmo di cui ALNS costituisce una generalizzazione, grazie all'uso ragionato di numerose euristiche di distruzione/ricostruzione.

Di seguito sono illustrati i punti cardine dell'algoritmo utilizzato. Dopo aver individuato la soluzione ammissibile iniziale tramite un algoritmo costruttivo, si passa alla fase di distruzione (ovvero inserimento) che consiste nell'aggiungere q cluster seguendo l'approccio dell'euristica scelta per quell'iterazione. Ad ogni passo iterativo, l'euristica di distruzione restituisce una soluzione tendenzialmente *infeasible*, ovvero che non rispetta la finestra temporale prefissata. Per riportare in campo di ammissibilità questa soluzione, vengono successivamente rimossi da un'euristica di riparazione (ovvero rimozione) un numero di cluster tali da arrivare a rispettare il vincolo di tempo. Ad ogni iterazione la scelta dell'euristica di inserimento e di rimozione è basata su un meccanismo di *Roulette wheel selection* che, a fronte dei risultati, premia le euristiche che hanno dimostrato una maggior efficacia nel corso della ricerca, rendendone più probabile la selezione.

La ALNS è una "metaeuristica inclusa in un'altra metaeuristica" che prevede l'applicazione della *Simulated Annealing* [5] (SA), nel nostro caso riadattata ad un problema di massimizzazione. SA prevede che, data la soluzione precedentemente accettata x , la soluzione ottenuta nell'iterazione corrente x^t e le loro rispettive funzioni obiettivo $c(x)$ e $c(x^t)$, la probabilità di accettazione della soluzione x^t sia:

$$e^{\frac{c(x^t) - c(x)}{T}} \quad (13)$$

dove T è la temperatura che viene decrementata ad ogni iterazione attraverso un parametro $\alpha \in [0,1]$. La probabilità di accettare soluzioni peggiorative si riduce al diminuire di T .

Al fine di controllare meglio l'andamento delle soluzioni individuate, si suddivide la ricerca in un certo numero di segmenti, ognuno composto da un numero fissato di iterazioni (selezionabile tramite il parametro "*Max Iterations per Segment*").

All'inizio di ogni segmento vengono effettuate le seguenti operazioni:

- La temperatura utilizzata dalla SA viene resettata al valore iniziale, che si è scelto essere il doppio della funzione obiettivo della soluzione ottima del rilassato.
- Al primo segmento, alle euristiche viene assegnato un peso unitario. Se il segmento non è il primo, i pesi delle euristiche migliori vengono impostati al valore "*Reward For Best*".

Segment Heuristics”, quelli delle euristiche peggiori vengono impostati a “*Punishment For Worst Segment Heuristics*”, mentre le altre euristiche tornano ad avere un peso unitario.

Alla fine di ogni segmento:

- Si applica una ricerca locale sulla soluzione migliore del segmento.

Uno dei pregi dell’ALNS risulta essere la diversificazione, infatti la soluzione risulta modificata del 30%/40% ad ogni singolo step. Per controbilanciare questo aspetto, una volta individuata una soluzione potenzialmente buona alla fine di ogni segmento, si opera una *local search* per intensificare la ricerca sfruttando la modellizzazione matematica e i vincoli euristiche aggiunti.

Infine, si è proposta una specie di memoria a lungo termine che tenesse conto, per ciascun cluster della probabilità di ottenere soluzioni migliori in termini di efficacia o soluzioni *infeasible*. Questo approccio evita di ricadere in cluster già visitati, diversificando la ricerca senza, però, limitarla.

3. MIGLIORAMENTO DELLA MODELLIZZAZIONE

Per migliorare la modellizzazione di partenza del problema sono stati aggiunti alcuni vincoli, che hanno permesso di ridurre i tempi computazionali, avvicinandosi al *convex hull*. Per quanto concerne i vincoli euristiche, questi vengono attivati in maniera euristica: considerando la soluzione iniziale, generata tramite algoritmo costruttivo, vengono testati i diversi vincoli in modo tale che solo quelli che vengono rispettati nella soluzione di partenza possano essere inseriti nella modellizzazione. Pur essendo un approccio molto restrittivo, i benefici in termini di tempo sono rilevanti in quanto il processo di *local search* è alleggerito.

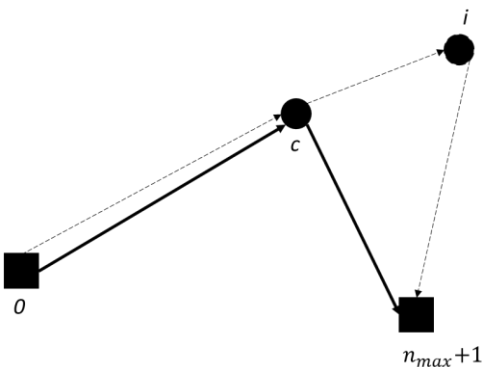
Vincoli esatti

- Prima dell'applicazione dell'algoritmo costruttivo, si eliminano i cluster c sicuramente *infeasible*, ossia si impone che:

$$\forall c \text{ tale che } \text{distanza}(0, c) + \text{distanza}(c, n_{max} + 1) + \text{durata servizi}(c) > t_{max}$$

$$y_c = 0, \quad x_{i,j}^v = 0 \quad \forall v \in V, \forall i, j \in c \quad (14)$$

Per la disuguaglianza triangolare il cammino diretto dal deposito di partenza, a un cluster al deposito di arrivo, incluso il tempo di servizio del cluster visitato, è sempre più breve di un qualunque cammino che includa anche un altro cluster. Di conseguenza per ogni cluster, se il suo costo di servizio più la distanza da entrambi i nodi di deposito è maggiore di t_{max} , tale cluster è escluso dal grafo perché è sicuramente *infeasible*.



$$\begin{cases} T_{0,c,n_{max}+1} = T_{0,c} + T_{servizio\ c} + T_{c,n_{max}+1} \\ T_{0,c,i,n_{max}+1} \geq T_{0,c,n_{max}+1} \\ T_{0,c,i,n_{max}+1} = T_{0,c} + T_{servizio\ c} + T_{c,i} + T_{servizio\ i} + T_{i,n_{max}+1} \end{cases}$$

- Tutti gli archi all’interno di un cluster, eccetto quelli specificati da un vincolo di precedenza, devono essere rimossi.

$$x_{i,j}^v = 0 \quad \forall v \in V, \forall c \in C, \forall i, j \in c : j \neq i, j \neq i + 1 \quad (15)$$

Vincoli Euristici

- Tutti gli archi che collegano il deposito iniziale ad altri nodi del cluster, diverso dal primo, devono essere rimossi. È un vincolo euristico perché esclude il caso particolare in cui un cluster, posizionato in coincidenza col deposito di partenza e dotato di servizi a costo nullo, richieda più veicoli distinti per essere servito.

$$x_{0,j}^v = 0 \quad \forall v \in V, \forall c \in C, \forall j \in c \setminus \{\text{nodo iniziale}\} \quad (16)$$

- Tutti gli archi che partono da un qualsiasi nodo di ogni cluster, escluso l'ultimo, e lo collegano al nodo di deposito finale devono essere rimossi.

$$x_{i, n_{max}+1}^v = 0 \quad \forall v \in V, \forall c \in C, \forall i \in c \setminus \{\text{nodo finale}\} \quad (17)$$

È un vincolo euristico molto forte, ma che può essere utile per ridurre la dimensione del problema in modo rapido.

- Il numero di cluster serviti deve essere almeno pari al numero di veicoli disponibili. Ogni veicolo che parte dal deposito all'istante $t=0$ deve aprire un nuovo cluster.

$$\sum_{c \in C} y_c \geq |V| \quad (18)$$

Si tratta di un vincolo euristico in quanto vi può essere comunque la possibilità che un veicolo, non riuscendo a svolgere alcun servizio a partire dal deposito iniziale, si diriga direttamente al deposito finale senza "aprire" alcun cluster.

- Analizzando il rilassamento continuo, si sono individuati dei casi in cui le variabili $x_{i,j}^v$ sono risultate diverse da 0, mentre le rispettive variabili $z_{i,j}$ erano uguali a 0. Si è introdotto quindi il seguente vincolo:

$$\sum_{v \in V} x_{i,j}^v \leq \frac{z_{i,j}}{\epsilon} \quad \text{con } \epsilon > 0 \text{ piccolo a piacere, fissato al valore di } OptimalityTol \text{ di Gurobi} \quad (19)$$

- Per l'implementazione dell'ultimo vincolo è stato necessario definire il concetto di *Streak*, ossia un insieme di nodi in sequenza S_v in un cluster che possono essere serviti tutti in una volta dal veicolo $v \in V$. Con il vincolo si impone che se un veicolo v entra in uno *Streak* S_v non può uscire finché non serve tutti i nodi appartenenti a S_v . Per fare ciò, per ogni cluster si identifica una lista ordinata di servizi richiesti e per ogni veicolo si cerca ogni *Streak* che può essere servito in quel particolare cluster. A questo punto, per ogni *Streak* di uno specifico veicolo si rimuovono tutti gli archi che vanno da un nodo (diverso da quello iniziale e quello finale), a un qualsiasi altro nodo, diverso dal successivo in base alle precedenze. Gli archi che incidono sul nodo iniziale e quelli che partono dal nodo finale di uno *Streak* rimangono inalterati.

4. ALGORITMO COSTRUTTIVO

L'algoritmo costruttivo ha l'obiettivo di produrre rapidamente una soluzione *feasible* per il problema. Per far ciò sono introdotti progressivamente clusters $c \in C$ fino a che il tempo necessario per raggiungerli e servirli non sfora il vincolo di tempo T_{max} .

Quando una soluzione con l'aggiunta di un cluster viola il vincolo, consideriamo come output dell'algoritmo la soluzione senza il cluster che la rende *infeasible*.

I clusters sono introdotti in soluzione secondo un ordine preciso finalizzato al raggiungimento di una soluzione abbastanza buona.

Tale ordine è raggiunto utilizzando tre criteri di ordinamento successivi:

- Ordinamento decrescente dei cluster in base al seguente rapporto:

$$\frac{\text{profitto}}{|\text{veicoli del cluster}| * \text{durata del servizio}} \quad (20)$$

Il numero dei veicoli utilizzati è il numero minimo di veicoli necessari a servire un cluster; la durata del servizio indica il tempo necessario a completare tutti i servizi nel cluster.

- Ordinamento crescente in base al numero dei nodi nei cluster.
- Ordinamento crescente in base al numero massimo di veicoli necessario a servire un cluster.

L'ultimo ordinamento applicato sarà quello con la priorità maggiore: introdurremo prima i cluster con il numero minore di veicoli massimi necessari per servirlo; a parità di numero massimo di veicoli, introdurremo un cluster con un numero minore di nodi; a parità di numero di nodi, conterà il profitto pesato.

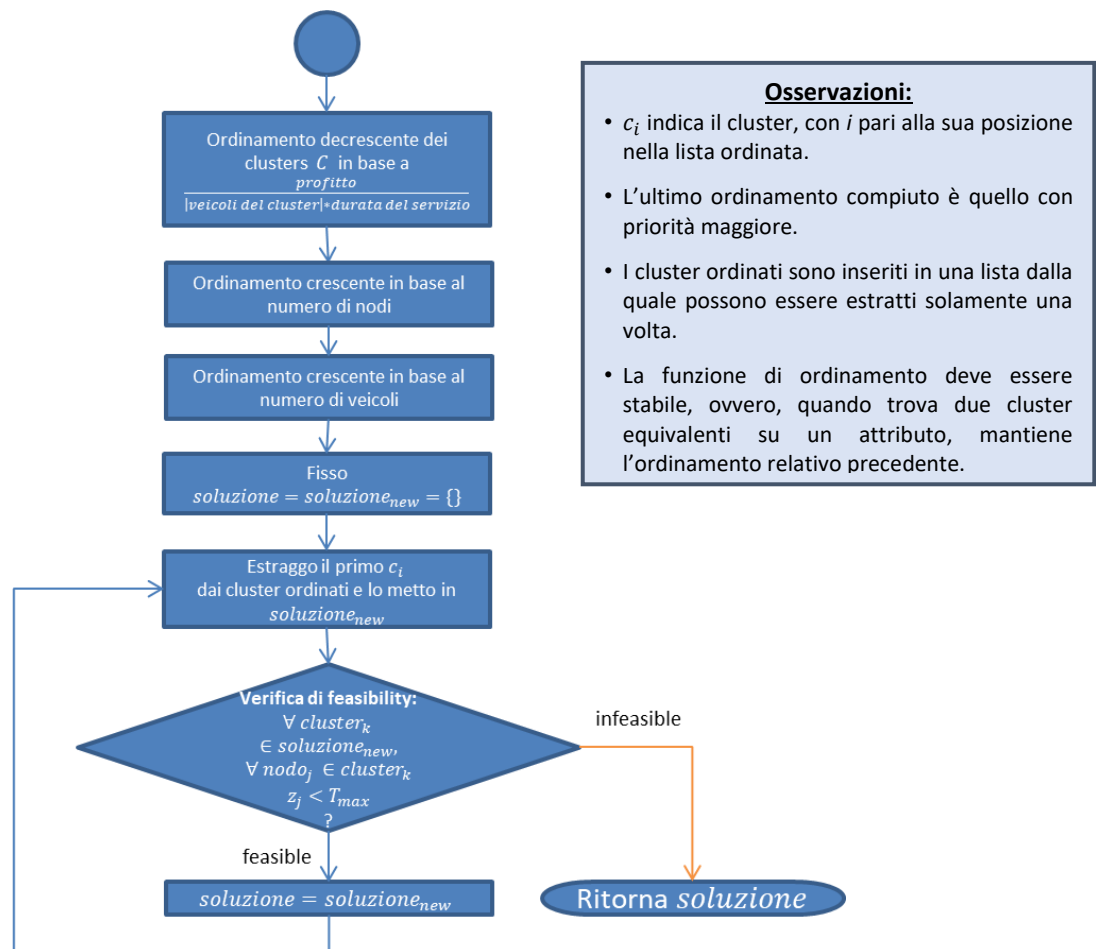


Figura 1: Flow Chart dell'algoritmo costruttivo

Ad esempio, si procederà inserendo in soluzione per primi tutti i cluster costituiti da un singolo nodo (e quindi serviti da un singolo veicolo), per poi passare ai cluster costituiti da due nodi, sempre serviti da un singolo veicolo. Nelle iterazioni successive si seguirà la stessa procedura aumentando il numero dei nodi dei cluster.

Una volta presi tutti i cluster con i nodi visitati da un singolo veicolo si passerà a quei cluster i cui nodi sono soddisfatti da due veicoli. Si ripeteranno i passi precedenti per i cluster con due veicoli. Dopo di che si passerà a cluster visitati da più di due veicoli, iterando sempre nello stesso modo. In *Figura 1* è riportato il *flow chart* con la rappresentazione grafica delle operazioni da eseguire per la creazione della soluzione iniziale.

5. METAEURISTICA

Alla base dell'algoritmo proposto si ha una ALNS, metaeuristica che ha l'obiettivo di trovare una soluzione migliore mediante un processo iterativo in cui ad ogni iterazione una parte della soluzione corrente viene distrutta per spostare lo spazio della ricerca.

Il modello tradizionale, proposto da *Ropke e Pisinger* [4], è stato adattato al nostro problema di massimizzazione considerando come euristiche distruttive quelle di inserimento di clusters in soluzione; viceversa sono considerate euristiche di riparazione quelle di rimozione dei clusters. La ragione di questa variazione è dovuta al fatto che le euristiche di inserimento portano le soluzioni verso l'*infeasibility*. L'inserimento di un numero eccessivo di cluster da visitare non consente infatti il soddisfacimento del vincolo di tempo.

In questa fase risulta fondamentale la definizione del parametro $q \in [q_{initial}, q_{max}]$, che rappresenta la dimensione dell'intorno e identifica, nel problema in esame, il numero di cluster da inserire dall'euristica di distruzione. Si è prediletto come componente il cluster perché consente un buon *trade off* tra efficacia ed efficienza: la scelta del singolo nodo, oltre ad aumentare la difficoltà computazionale, non avrebbe consentito di controllare l'andamento della ricerca.

5.1 Principali Parametri

Il parametro q rappresenta il numero di cluster rimossi dall'euristica di rimozione. Inizialmente è fissato $q = q_{initial}$ e viene incrementato nel corso della ricerca fino ad un valore q_{max} pari al numero totale di cluster nell'istanza data. Alla fine di ogni segmento si amplia l'intorno analizzato grazie ad un parametro Δq . Ciò comporta un aumento della difficoltà computazionale che viene tenuta sotto controllo anche grazie all'introduzione di una memoria di medio periodo che tenga conto delle prestazioni di ciascun cluster nelle soluzioni (Vedi *Capitolo 6 - Cluster Roulette*).

Si è implementato un andamento di q circolare: al termine di un segmento, q viene aggiornato secondo la seguente relazione:

$$q_{new} = \max((q_{old} + \Delta q) \bmod q_{max}, q_{min}) \quad \text{dove } q_{min} = 1 \quad (21)$$

Si prende il massimo fra i due valori in quanto può capitare che $(q_{old} + \Delta q) \bmod q_{max}$ sia pari a zero: ciò avviene quando $q_{old} + \Delta q$ risulta essere un multiplo intero di q_{max} .

A questo punto, quindi, risulta indispensabile porre limiti alla lunghezza del segmento. Con il parametro *SegmentSize* viene tarato il numero massimo di iterazioni per ciascun segmento. Per controllare meglio l'andamento della ricerca si introduce *maxSegmentsWithoutImprovement*, che rappresenta il numero massimo di iterazioni senza miglioramento che possono essere accettate prima di passare al successivo segmento. L'introduzione di questo criterio d'arresto

consente un'esplorazione più efficiente delle soluzioni: se l'algoritmo si blocca in una lunga serie di iterazioni senza miglioramento, allora si sposta l'attenzione su un'area più vasta dello spazio della ricerca dove, potenzialmente, sarà più probabile individuare soluzioni migliori.

5.2 Euristiche di Distruzione

In letteratura sono presenti molte euristiche di inserimento, alcune delle quali sono state riprese per essere adattate al problema in esame. Inoltre, la presenza dei cluster e dei veicoli ha permesso di introdurre altre completamente innovative.

Per ciascuna euristica sono stati considerati i parametri *InputSolution*, che costituisce la soluzione di partenza, q , che indica il numero di cluster da inserire, ed infine *Output*, che rappresenta la soluzione ottenuta.

Di seguito sono riportate le euristiche di distruzione utilizzate:

Best Insertion: q cluster con il rapporto profitto/costo (somma delle durate dei servizi) maggiore sono inseriti nella soluzione corrente in maniera *greedy*.

Essa consente un buon compromesso tra efficacia, intesa come profitto associato ad un cluster, ed efficienza, intesa come somma delle durate dei servizi da svolgere presso un cluster.

Cost Insertion: q cluster con il minor costo (somma delle durate dei servizi) sono inseriti nella soluzione corrente in maniera *greedy*.

Profit Insertion: q cluster con il maggior profitto sono inseriti nella soluzione corrente in maniera *greedy*.

Random Insertion: q cluster sono inseriti, in modo casuale, nella soluzione corrente in maniera *greedy*.

Close to Barycenter Insertion: q cluster più vicini al baricentro sono inseriti nella soluzione corrente in maniera *greedy*. Il baricentro è calcolato sulla soluzione input di partenza ed ha quindi il vantaggio di evitare la dispersione dei veicoli in punti troppo lontani da quelli fin ora visitati. È un'euristica che consente anche di risolvere, in parte, il problema della generazione di soluzioni *infeasible* causate dall'inserimento di un cluster così lontano da richiedere la rimozione di un numero eccessivo di cluster per soddisfare i vincoli di tempo.

5.3 Euristiche di Riparazione

Dopo l'applicazione delle euristiche di inserimento, si procede con quelle di rimozione, togliendo dalla soluzione un numero di cluster θ scelti tra l'insieme dei cluster disponibili e tali da ritornare in condizioni di ammissibilità.

Come nel caso delle euristiche di inserimento, sono stati considerati i parametri *InputSolution* ed *Output*.

Di seguito sono riportate le euristiche di riparazione utilizzate:

Worst Removal: Rimuove dalla soluzione i θ cluster con il minor rapporto profitto/costo in maniera *greedy*. Tale euristica, come la "*Best Insertion*", consente di ottenere un buon *trade off* tra efficacia ed efficienza.

High Cost Removal: Rimuove dalla soluzione i θ cluster con il costo (somma delle durate dei servizi) più alto in maniera *greedy*.

Random Removal: Rimuove dalla soluzione, in modo casuale, θ cluster in maniera *greedy*.

Travel Time: Rimuove il primo cluster i con il rapporto profitto/costo minore, poi rimuove i $\theta - 1$ cluster che sono più simili al primo scelto.

La misura di relatività è così definita:

$$\frac{1}{3}(d(i, j) + d(sorgente, j) + d(j, terminale)) \quad (22)$$

Dove:

$$\begin{cases} i = \text{primo cluster rimosso dalla soluzione} \\ j = \text{cluster su cui valutare la rimozione} \\ d(x, y) = \text{distanza tra } x \text{ e } y \end{cases}$$

Rispetto ad una semplice “*Worst Removal*”, oltre a considerare la durata del servizio, si considera anche la distanza euclidea che è una componente molto importante nella verifica del soddisfacimento del vincolo di tempo. Con questa euristica si vuole eliminare innanzitutto il cluster con minor beneficio in base alla durata e si vogliono rimuovere quei cluster simili in termini di distanza al fine di riottimizzare completamente il percorso dei veicoli in quella zona.

Vehicle Time: Rimuove il primo cluster i con il rapporto profitto/costo minore, poi rimuove i $\theta - 1$ cluster che sono più simili al primo scelto.

La misura di relatività è così definita:

- Trovare il veicolo $v \in V$ con il tempo di servizio più lungo nel primo cluster i .
- Per ogni nodo n di un cluster generico definire la durata del servizio che può effettuare v e sommare tutti i valori.
- Dividere il valore appena calcolato per la durata totale di tutti i servizi nel cluster in esame.
- Prendere il valore assoluto della differenza tra il rapporto del primo cluster i rimosso e il rapporto di tutti gli altri cluster come criterio di similarità.

Questa euristica consente di riottimizzare il percorso del veicolo v che persisteva per maggior tempo sul primo cluster i rimosso. Eliminando i cluster con rapporto simile, si fornisce all’algoritmo più libertà per l’allocazione ottimale del veicolo v in esame, definendo però un logico criterio di rimozione.

5.4 Assegnazione e Aggiornamento dei Pesi

Poiché l’ALNS opera su un set di euristiche, è necessario definire un sistema di selezione e valutazione dei punteggi da assegnare a ciascuna euristica disponibile. Si è implementata una procedura adattiva di assegnamento dei punteggi per rappresentare l’effettiva *performance* dell’operatore in esame.

Ad ogni euristica j è assegnato un punteggio P_j , inizialmente fissato a 1. Considerando che ogni euristica di rimozione è selezionata indipendentemente da quella di inserimento (e viceversa), ad ogni iterazione si aggiornano i valori in base a quattro casistiche principali:

$$\Psi = \max \begin{cases} \omega_{j,1} & \text{se la nuova soluzione è un } \textit{ottimo globale} \\ \omega_{j,2} & \text{se la nuova soluzione è } \textit{migliore} \text{ di quella corrente} \\ \omega_{j,3} & \text{se la soluzione } \textit{non} \text{ è migliore di quella corrente, ma è comunque } \textit{accettata} \\ \omega_{j,4} & \text{se la soluzione } \textit{non} \text{ è migliore di quella corrente e viene } \textit{rifiutata} \end{cases}$$

Il peso ω_j è un valore maggiore di zero per evitare che un’euristica abbia una probabilità di estrazione nulla.

Ad ogni iterazione, il punteggio dell’euristica selezionata viene aggiornato secondo la formula:

$$P_{\text{new}} = \lambda P_{\text{old}} + (1 - \lambda) \Psi \quad (23)$$

Dove $\lambda \in [0,1]$ è un parametro di decadimento che rappresenta la velocità di variazione della probabilità: maggiore è λ , maggiore è l'importanza assegnata al risultato dell'ultima iterazione. Si noti che questa formula non esclude la possibilità che la probabilità associata ad un'euristica j diminuisca nel corso della ricerca.

All'inizio di ogni segmento i punteggi P_j vengono fissati nuovamente a 1, perché ci si è spostati in una zona dello spazio delle soluzioni potenzialmente diversa da quella precedente ed è quindi necessario avere a disposizione tutte le euristiche, al fine di garantire un'esplorazione adeguata. Per consentire un miglior controllo dell'andamento della ricerca, si è preferito però tener conto dell'euristica considerata migliore e di quella considerata peggiore nel segmento precedente aggiornandone il punteggio:

$$P_{new_best} = Reward * P_{old_best} \quad (24)$$

$$P_{new_worst} = Punishment * P_{old_worst} \quad (25)$$

6. CLUSTER ROULETTE

Come riportato nel *Capitolo 2 - Soluzione proposta*, l'algoritmo ha come punto di partenza una ALNS che viene arricchita e modificata prendendo spunto da altre metaeuristiche. Uno dei principali problemi riscontrati nell'analisi delle soluzioni individuate, risulta essere che una semplice ALNS, pur diversificando, rischia di ciclare, non spostandosi da soluzioni *infeasible*.

Per questo motivo, si è implementato un meccanismo detto *Cluster Roulette*, ispirato ad una *Tabu Search*, che permette di limitare il ripresentarsi di soluzioni già viste o non ammissibili, attribuendo una certa probabilità di selezione ai cluster disponibili per le euristiche di inserimento dell'ALNS.

Tale probabilità di selezione è inizialmente posta a 1 per ogni cluster, ma viene variata ad ogni iterazione di ALNS mediante i seguenti meccanismi base di combinazione convessa:

1. Si aumenta la probabilità di selezione grazie ad un fattore γ_{up}

$$P_{new} = \gamma_{up} * P_{old} + (1 - \gamma_{up}) \quad (26)$$

2. Si diminuisce la probabilità di selezione grazie ad un fattore γ_{down}

$$P_{new} = \gamma_{down} * P_{old} \quad \text{oppure} \quad P_{new} = (1 - \gamma_{cooldown}) * P_{old} \quad (27)$$

All'inizio di ogni euristica di inserimento/distruzione sulla soluzione iniziale x_{old} , viene stilata una lista di clusters non facenti parte della soluzione iniziale; dunque ciascuno dei cluster disponibili è reso disponibile all'euristica di inserimento secondo la sua probabilità di selezione corrente.

Alla fine del test di *feasibility* sulla soluzione prodotta dall'euristica di inserimento, le probabilità di selezione variano seguendo i seguenti meccanismi:

Cooldown: i cluster che sono appena stati selezionati dall'euristica di inserimento (hot cluster) avranno una probabilità di selezione inferiore nelle successive iterazioni. Anche i cluster rimossi dall'euristica di rimozione/riparazione subiscono il medesimo trattamento.

$$P_{new} = (1 - \gamma_{cooldown}) * P_{old} \quad \gamma_{cooldown} \in [0,1] \quad (28)$$

Warmup: i cluster che non sono stati appena selezionati dall'euristica di inserimento (cold cluster) avranno una probabilità di selezione superiore nelle successive iterazioni.

$$P_{new} = \gamma_{warmup} * P_{old} + (1 - \gamma_{warmup}) \quad \gamma_{warmup} \in [0,1] \quad (29)$$

Punishment: i cluster che si sono comportati male in un segmento, anche determinando frequenti soluzioni *infeasible* (*nerf candidates*¹) avranno una probabilità iniziale di selezione molto inferiore nel segmento successivo, determinata così:

$$P_{\text{new}} = \gamma_{\text{punishment}} \quad \gamma_{\text{punishment}} \in [0,1] \quad (30)$$

Si definisce *NerfBarrier* $\in [0,1]$ la percentuale di tempo (misurato come numero di aggiornamenti di probabilità della cluster roulette) entro cui la probabilità di selezione di un cluster deve essere inferiore alla media per essere punito ulteriormente attraverso il fattore $\gamma_{\text{punishment}}$. Un cluster la cui probabilità di selezione è sotto la media delle probabilità di tutti i cluster per una quantità di tempo superiore a *NerfBarrier* è detto *nerf candidate*.

Per ridurre la complessità della *local search*, si impone anche che i *nerf candidates* (purché non facciano parte della soluzione migliore corrente) vengano esclusi dai cluster disponibili per la *local search* al fine di aumentare l'efficienza del modello.

Come per i pesi assegnati alle singole euristiche, anche le probabilità di selezione dei clusters sono riportate a 1 a fine segmento, eccetto per i *nerf candidates*, la cui probabilità di selezione sarà fissata a $\gamma_{\text{punishment}}$.

7. SIMULATED ANNEALING

L'algoritmo implementato, come già anticipato, incorpora caratteristiche che permettono di migliorarne le performance, come la *Simuated Annealing* (Kirkpatrick et al. 1983 [5]). Alla fine di ogni iterazione si è deciso di applicare un criterio di accettazione per ognuna delle soluzioni identificate dall'esecuzione dell'ALNS. Se la nuova soluzione soddisfa il sopracitato criterio di accettazione, allora questa sarà utilizzata come soluzione di partenza per l'iterazione successiva. Tipicamente, in un problema di massimo, la nuova soluzione x^t sostituisce la soluzione precedente x se $c(x^t) - c(x)$, dove $c(x)$ rappresenta il valore della soluzione x . Utilizzando, invece, il paradigma di scelta della *Simulated Annealing* si ha che, definito $\Delta c = c(x^t) - c(x)$, x' verrà accettato con una probabilità pari a:

$$e^{\frac{c(x^t) - c(x)}{T}} \quad (31)$$

dove $T > 0$ è la temperatura.

In un problema di massimo, si possono verificare due casi:

- Se la soluzione trovata è migliore di quella precedente $c(x^t) > c(x)$ allora l'esponente è positivo e quindi la probabilità di selezione è maggiore o uguale a 1 per qualsiasi valore di T .
- Se la soluzione trovata è peggiore di quella precedente $c(x^t) < c(x)$ allora l'esponente è negativo e quindi la probabilità di selezione è inferiore a 1. In funzione del Δc e della temperatura varierà la probabilità di accettazione.

¹ Solitamente si parla di [nerfing](#) nel gergo videoludico per riferirsi a depotenziamenti dell'equipaggiamento di un giocatore, introdotti dagli sviluppatori per scoraggiarne l'uso. Tali misure sono spesso introdotte negli aggiornamenti dei videogiochi per equilibrare la *fairness* dell'esperienza multiplayer, specialmente quando il ventaglio di equipaggiamenti selezionabile da un giocatore è molto ampio, molto variabile e dunque non è facile per gli sviluppatori capire da subito se un elemento porterà giocatori a sopraffare degli altri per ragioni non correlate alla loro abilità di gioco, generando scontento nella comunità.

Al termine di ogni iterazione verrà aggiornata la temperatura:

$$T_{new} = \alpha \cdot T_{old} \quad \alpha \in [0,1] \quad (32)$$

Il parametro α rappresenta la velocità di decrescita della temperatura: al diminuire di T la probabilità di accettazione della soluzione x^t diminuisce. Il parametro α è fondamentale nella SA e deve essere ben tarato:

- Un α troppo grande comporta una ripida discesa della temperatura, ostacolando l'accettazione di soluzioni peggiorative che consentirebbero di spostarsi in un nuovo spazio della ricerca;
- Un α troppo piccolo comporta l'accettazione di tutte le soluzioni peggiorative, impendendo di intensificare la ricerca in aree potenzialmente molto buone.

Per avere un valore comparabile con la differenza della funzione obiettivo ($c(x^t) - c(x)$) si è fissata come temperatura iniziale un valore pari al doppio della funzione obiettivo della soluzione ottima del rilassamento continuo. All'inizio di ogni segmento la temperatura viene rifissata al valore iniziale, perché, essendosi spostati in un nuovo spazio della ricerca, è necessario garantire la possibilità di accettare soluzioni peggiorative in modo da uscire da un eventuale ottimo locale.

8. LOCAL SEARCH

Alla fine di ogni segmento, una procedura di *Local Search* (LS) è applicata sulla migliore soluzione *feasible* individuata durante il segmento. L'obiettivo è di intensificare la ricerca, allontanandosi da un classico meccanismo di rimozione e inserimento, tipico dell'ALNS.

A tale scopo si è deciso di sfruttare la modellizzazione matematica di partenza aggiungendo vincoli esatti ed euristici (*Capitolo 3 - Miglioramento della modellizzazione*) che permettessero di raggiungere il più velocemente possibile soluzioni potenzialmente buone.

9. TARATURA DEI PARAMETRI

Per rendere la modellizzazione più vicina al problema di *Team Orienteering* sono stati definiti alcuni parametri ritenuti rilevanti. Durante l'implementazione dell'algoritmo, ha avuto inizio un primo processo di taratura preliminare dei parametri, tenendo conto degli effetti reciproci. Nella *Tabella 1* seguente, sono rappresentati i valori inizialmente fissati sulla base di un'analisi in itinere.

Parametro	Valore	Parametro	Valore
Initial q	1	Nerf Barrier	0,38
Delta q	2	Local search time limit	200
Max Iterations per segment	50	Max MIPS nodes to solve in feasibility check	10000
Max Iterations w/o improvement in a segment	16	Heuristics scores w1	2
Lambda	0,7	Heuristics scores w2	1,5
Alpha	0,7	Heuristics scores w3	0,5
Punishment Gamma	0,6	Heuristics scores w4	0,1
Cooldown Gamma	0,05	Reward for best segment heuristics	1,4
Warmup Gamma	0,005	Punishment for worst segment heuristics	0,6

Tabella 1- Valori individuati tramite processo di taratura preliminare dei parametri

Una volta terminata la stesura del codice, grazie all'interfaccia (*Figura 2*) sviluppata da F. Piazza, è stato possibile realizzare una taratura migliore con un'analisi più puntuale in modo da avvicinarsi il più possibile alla soluzione ottima, nel minor tempo computazionale. Per ciascun parametro sono stati valutati tre o due possibili valori su un tempo di 600 secondi, evidenziando i risultati migliori per ciascuna istanza. I parametri migliori in ciascuna istanza sono stati analizzati su un foglio di lavoro di Excel, dopo essere stati testati, per tre volte, per 30 minuti.

La scelta di testare le istanze per 10 minuti è nata dal compromesso tra efficacia ed efficienza.

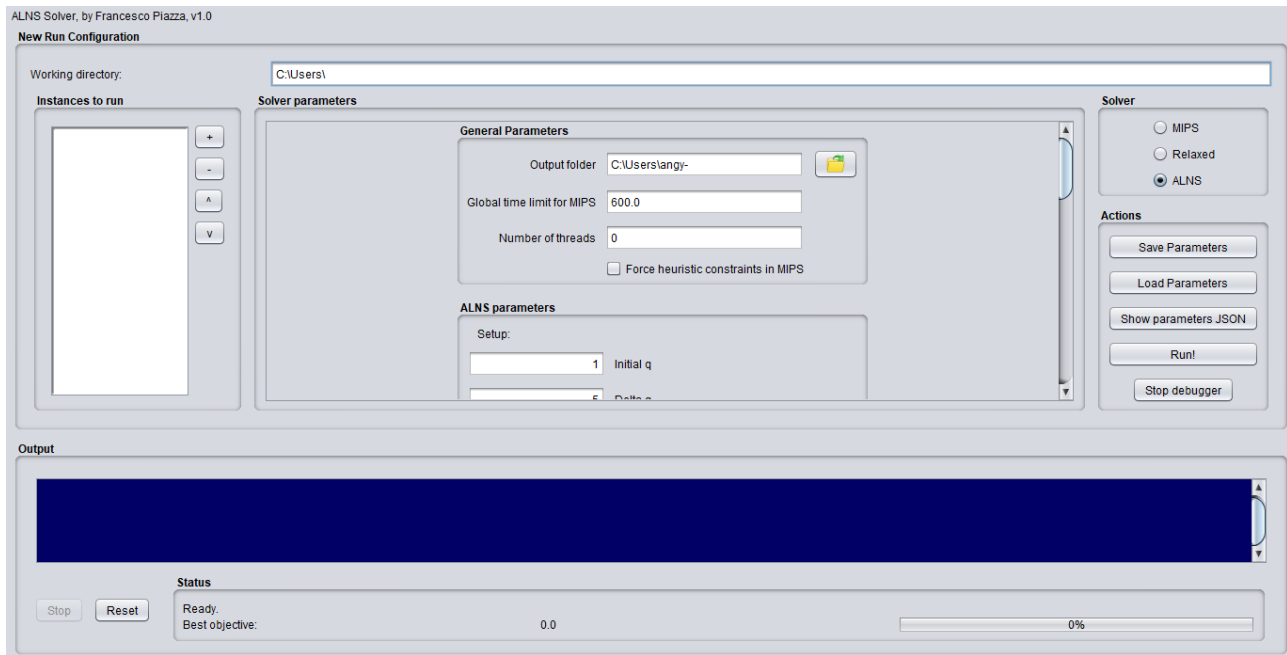


Figura 2 – Instantanea dell'interfaccia utilizzata per la parametrizzazione

9.1 Initial q

Valore di q (grado di distruzione) all'inizio del primo segmento della ALNS. Il parametro q determina quanti cluster devono essere inseriti da un'euristica di distruzione e il valore corrispondente cresce alla fine di ogni segmento di un valore "Delta q ".

Parametrizzazione

Il processo di parametrizzazione ha un ordine ben preciso: dapprima sono stati analizzati i parametri più indipendenti, sui quali si basa il funzionamento complessivo dell'algoritmo, in seguito l'attenzione si è concentrata sui parametri più specifici tenendo conto di eventuali rapporti reciproci e dei parametri fissati in precedenza.

Per tale motivo, il primo parametro ad essere stato tarato è *Initial q* che è alla base di tutta la metaeuristica.

Per ogni istanza sono stati testati differenti valori della variabile *Initial q*: il valore base, che corrisponde ad 1, ed altri valori che variavano a seconda del numero di cluster presenti in ogni istanza. In particolare, si è considerato il parametro in base al 10%, 25% e 50% rispetto al numero di cluster presenti nell'istanza. La parametrizzazione in funzione del numero di cluster consente di differenziare *Initial q* per la specifica istanza: all'aumentare del numero di cluster potrebbe essere utile aumentare il valore del parametro in modo tale da spaziare nella ricerca, iniziando con un *Initial q* significativo.

Dopo aver analizzato ciascuna istanza è stato possibile trovare il valore ottimale per ognuna di esse. Nell'80% delle istanze risulta essere migliore il valore base di q , mentre nel restante 20% è risultato migliore il parametro corrispondente al 10% del numero di cluster. Il miglioramento medio derivato dalla variazione del parametro *Initial q* è risultato pari al 55%.

9.2 Delta q

Il parametro *Delta q* indica l'incremento costante di *q* alla fine di ogni segmento.

Parametrizzazione

Questo parametro inizialmente è stato valutato come pari a 2. Per valutare il comportamento delle istanze al variare del parametro, esso è stato modificato in due differenti modi:

- Valore unitario
- $Deltaq = \frac{Initial\ q}{(1+\%rispettoc)}$

dove *%rispettoc* risulta essere la percentuale di Initial *q* migliore rispetto al numero di cluster totali scelti in soluzione.

Nel caso in cui Initial *q* migliore fosse quello base, allora *Delta q* era pari esattamente a Initial *q*.

La scelta di tale formula deriva dal fatto che se si parte con un Initial *q* alto, il tempo computazionale e l'intorno da analizzare sono già molto ampi, quindi si potrebbe preferire effettuare incrementi non troppo elevati, per non rendere il problema troppo complesso. Al contrario se si parte da un intorno piccolo allora lo scopo è di diversificare, cercando di spaziare verso intorni più grandi. Il valore unitario è stato introdotto per garantire una completa analisi rispetto al valore base, ossia per valutare l'effetto nel caso di *Delta q* inferiore al valore base.

Per il 73,3% delle istanze il valore migliore di funzione obiettivo si è ottenuto mantenendo il parametro invariato, ossia pari a 2. Per tre delle restanti istanze il valore migliore è risultato *Delta q* pari ad uno, mentre per una sola è risultato pari a 7. Si tenga conto che, per queste quattro istanze, il valore di *Delta q* migliore corrisponde a quello trovato con la formula in funzione di Initial *q*: ciò conferma le ipotesi definite in precedenza.

Max Iterations per segment

Il parametro indica il massimo numero di iterazioni in un segmento di ottimizzazione di ALNS. Ogni iterazione include l'applicazione di un'euristica di distruzione sulla precedente soluzione *feasible*, seguita dall'applicazione di un'euristica di riparazione per riportare la soluzione distrutta alla *feasibility*.

Il parametro non è stato ritenuto significativo e quindi non è stato analizzato in modo approfondito. Si è preferito agire sul parametro successivo, che influenza direttamente la lunghezza dei segmenti.

9.3 Max Iterations w/o improvement in a segment

Il parametro indica il massimo numero di iterazioni senza alcun miglioramento in un segmento di ottimizzazione di ALNS. Se questo numero di iterazioni senza miglioramenti è raggiunto, il risolutore della ALNS si sposta sul segmento successivo (eventualmente facendo una piccola ricerca locale tra i due segmenti).

Parametrizzazione

Il valore di partenza è stato posto uguale a 16, mentre poi è stato variato in funzione dei cluster mancanti da analizzare *w* calcolato come:

$$w = arr.ecc \left(\frac{\text{numero di custer} - Initialq}{Deltaq} \right)$$

Si è poi calcolata la percentuale (%) di iterazioni in un segmento che rappresenta il valore del *Max Iterations w/o improvement in a segment*:

$$\% = \frac{60 * 30}{w * 50 * 20}$$

Nel caso in cui il risultato sia maggiore di 1 il valore viene fissato quindi a 50.

La decisione è dovuta al fatto che dalle soluzioni precedenti si è visto che un'iterazione dura circa 20 secondi (numero calcolato per eccesso) e quindi l'analisi di un segmento richiede 20×50 secondi. Avendo a disposizione 60×30 (in mezz'ora) secondi, % rappresenta il rapporto tra il tempo disponibile e il tempo necessario per analizzare in modo completo tutti i cluster mancanti da analizzare. Maggiore è %, maggiore è il tempo a disposizione per analizzare il segmento. Per semplicità non si è tenuto conto del tempo speso per la *Local Search*.

Questo valore percentuale viene moltiplicato per il numero di iterazioni in un segmento (50) e duplicato per individuare un valore da testare di *Max Iterations w/o improvement in a segment*. La scelta di moltiplicare il valore per 2 è dettata dal fatto che il valore percentuale risultava essere così piccolo da non consentire un'analisi accettabile del segmento.

Per individuare l'ultimo valore da testare, si è deciso di intensificare la ricerca sull'istanza che, in quel momento, risultava essere la peggiore rispetto al *benchmark* (istanze 6.2.n -40% dal *benchmark*). Dopo aver effettuato diverse prove, si è individuato come valore ottimale 5 interazioni massime senza miglioramento. Questo numero è stato esteso anche alle altre istanze, portando ad un miglioramento in altre due istanze.

Nel 40% delle istanze si è ottenuto un miglioramento adottando un diverso valore rispetto a quello di base. In particolare, il ragionamento in funzione dei cluster mancanti w è risultato essere adatto per tre istanze ottenendo, per queste, una crescita media percentuale rispetto al *benchmark* dal -2% al +5%. L'analisi puntuale condotta sull'istanza 6.2.n per avvicinarla all'ottimo, che ha restituito un valore del *Max Iterations w/o improvement in a segment* pari a 5, ha ottenuto buoni risultati anche in altre due istanze, con una crescita media percentuale rispetto al *benchmark* dal +75% al +100%.

9.4 Lambda

È il parametro di decadimento per l'aggiornamento dei pesi dell'euristica, è compreso tra i valori 0 e 1. I pesi delle euristiche sono aggiornati alla fine di ogni iterazione secondo la seguente formula:

$$P_{\text{new}} = \lambda P_{\text{old}} + (1 - \lambda) \Psi$$

dove Ψ è il punteggio assegnato all'euristica.

Parametrizzazione

Il valore base fissato attraverso un'analisi preliminare, è stato posto pari a 0,7. Tale valore tende a rallentare il processo di aggiornamento dei punteggi, favorendo P_{old} .

Come già specificato, l'ordine con cui sono stati tarati i diversi parametri non è casuale: in questo caso si è notato un legame reciproco tra *Lambda* e *Max Iterations w/o improvement in a segment*. Minore è il numero di iterazioni analizzabili per ciascuno, maggiore deve essere la velocità di aggiornamento dei punteggi relativi alle euristiche. All'inizio di ogni segmento il parametro *Lambda* è reimpostato a 1 per ridurre l'influenza tra i diversi segmenti, garantendo una continua efficacia di analisi. Per questo motivo la sua parametrizzazione è stata fatta in funzione di *Max Iterations w/o improvement in a segment*:

- Si è deciso che nel caso in cui *Max Iterations w/o improvement* fosse stato maggiore o uguale a 16 (valore base), allora per assumere un decadimento lento si impostava un valore di *Lambda* pari a 0,85. Altrimenti *Lambda* era pari a 0,5.
- Tenendo conto che, comunque, un valore troppo elevato di λ può pregiudicare le prestazioni dell'algoritmo, impedendogli le applicazioni di euristiche effettivamente utili, si è scelto di testare il valore di 0,5 anche nel caso di un decadimento più lento. Per quanto riguarda il caso in cui, invece, *Max Iterations w/o improvement* fosse stato minore di 16, si è esasperata la velocità di aggiornamento con un valore pari a 0,3.

Per il 40% delle istanze il processo di *Tuning* dei parametri ha portato a dei miglioramenti medi del 39% rispetto al valore base: tre istanze ottengono risultati migliori con $\lambda = 0,5$ invece, due istanze con $\lambda = 0,3$ e infine una sola istanza con $\lambda = 0,85$. I vantaggi della parametrizzazione di *Lambda* sono i maggiori ottenuti in tutto il processo di analisi.

9.5 Alpha

È il parametro di decadimento per la temperatura nella *Simulated Annealing* ed è compreso tra i valori 0 e 1. Tale parametro è aggiornato alla fine di ogni segmento secondo la relazione:

$$T_{\text{new}} = \alpha T_{\text{old}}.$$

Una temperatura che decresce lentamente accetta peggiori soluzioni all'inizio del segmento, permettendo di trovare una soluzione migliore. Tuttavia, una temperatura che decresce in modo lento ci mette molto tempo a convergere ad una soluzione migliore.

Parametrizzazione

Il valore base fissato attraverso un'analisi preliminare è stato posto pari a 0,7 che rappresenta un decadimento lento di *Alpha*. Il processo di parametrizzazione di questo parametro è del tutto identico a quello del *Lambda* perché il principio alla base è lo stesso: minore è il numero di interazioni analizzabili in un segmento, maggiore deve essere la velocità di aggiornamento di α per accettare soluzioni peggiorative all'inizio della ricerca e in seguito intensificare l'analisi sulle soluzioni potenzialmente migliori. Si ricorda che anche questo parametro viene reimpostato a 1 all'inizio di un segmento. La parametrizzazione è avvenuta in funzione di *Max Iterations w/o improvement in a segment*:

- $\begin{cases} \text{Max Iterations}^{w/o \text{ improvement}} \geq 16 & \rightarrow \alpha = 0,85 \\ \text{Max Iterations}^{w/o \text{ improvement}} < 16 & \rightarrow \alpha = 0,50 \end{cases}$
- $\begin{cases} \text{Max Iterations}^{w/o \text{ improvement}} \geq 16 & \rightarrow \alpha = 0,50 \\ \text{Max Iterations}^{w/o \text{ improvement}} < 16 & \rightarrow \alpha = 0,30 \end{cases}$

In questo caso il processo di parametrizzazione non ha portato ad ulteriori miglioramenti rispetto al valore iniziale.

Punishment Gamma

Questo parametro serve per far sì che le euristiche di inserimento scelgano anche i cluster peggiorativi (*infeasible*). È una variabile compresa tra i valori 0 e 1.

La probabilità con cui i cluster peggiori vengono selezionati è:

$$P_{\text{new}} = (\text{PunishmentGamma})P_{\text{old}}$$

Il parametro *PunishmentGamma* rappresenta quale percentuale della probabilità precedente di estrazione rimane valida (ad esempio 0.01 = dopo la punizione resterà, per il cluster punito, l'1% della probabilità precedente di estrazione).

9.6 Cooldown Gamma

È il fattore che serve per diminuire la probabilità che un cluster venga rimosso/inserito nella soluzione. È compreso tra 0 e 1.

Un cluster caldo (appena selezionato) avrà una probabilità di essere scelto *CooldownGamma* volte minore della precedente, secondo la formula:

$$P_{\text{new}} = (1 - \text{CooldownGamma})P_{\text{old}}$$

Il parametro *CooldownGamma* determina di quanti punti percentuali deve essere più piccola la percentuale, ad esempio 0.05 = 5% più piccola.

Parametrizzazione

Il valore base ricavato dall'analisi preliminare è pari a 0,05. Si tratta di un parametro che, se ben utilizzato, consente di diversificare la ricerca evitando la selezione di cluster potenzialmente cattivi o evitando la ripetizione di soluzioni già viste. Tenendo conto di questa sua funzionalità, il processo di *Tuning* è stato svolto in funzione del miglioramento medio rispetto al benchmark fino ad ora ottenuto. Il ragionamento alla base era che se si era già giunti a una soluzione potenzialmente buona, l'obiettivo era di intensificare la ricerca ripescando anche soluzioni già viste; viceversa in caso di istanze ancora troppo vicine al risultato del benchmark, l'obiettivo era di diversificare per spostarsi in nuovi spazi di ricerca e quindi in nuove soluzioni.

- Nel caso in cui la soluzione ottenuta con il valore base ottenesse un miglioramento rispetto al benchmark inferiore rispetto a quello medio, allora per diversificare, *CooldownGamma* era raddoppiato. Viceversa, in caso di intensificazione il valore del parametro era dimezzato rispetto al valore base.
- Si è poi deciso di esasperare il ragionamento effettuato, raddoppiando o dimezzando il valore del *CooldownGamma* precedentemente individuato, in base alla casistica precedente.

Per il 73,33% delle istanze il valore base è stato riconfermato come migliore. La seconda alternativa, che consiste nell'esasperare il valore individuato in funzione del miglioramento medio, non risulta essere mai selezionato. In generale si ottiene un miglioramento del 5% rispetto alla configurazione precedente.

9.7 Warmup Gamma

È il fattore che serve per aumentare la probabilità che un cluster venga rimosso/inserito nella soluzione. È compreso tra 0 e 1.

Un cluster freddo (non selezionato di recente) avrà una probabilità di essere scelto *WarmupGamma* volte maggiore della precedente, secondo la formula:

$$P_{\text{new}} = (1 - \text{WarmupGamma})P_{\text{old}} + \text{WarmupGamma}$$

Il parametro *WarmupGamma* determina di quanti punti percentuali deve essere più grande la percentuale, ad esempio 0.05 = 5% più grande.

Parametrizzazione

Il valore base fissato attraverso un'analisi preliminare è stato posto pari a 0,005 che agisce sulla possibilità di operare un'intensificazione delle soluzioni. Il processo di parametrizzazione di questo parametro è esattamente il duale rispetto a quello del *CooldownGamma* perché, mentre il parametro precedente attua una diversificazione, questo attua un'intensificazione. Nel caso in cui la soluzione trovata è lontano da una soglia limite, soggettivamente fissata, allora l'obiettivo è di ridurre l'intensificazione e quindi ridurre *WarmupGamma*; viceversa l'obiettivo è di intensificare. Come soglia limite si è scelto il miglioramento medio percentuale rispetto al benchmark ottenuto fino alla parametrizzazione di *Alpha*.

- $\begin{cases} \% \text{ istanza (valore base)} \geq \% \text{ miglioramento medio} & \rightarrow \alpha = 0,01 \\ \% \text{ istanza (valore base)} < \% \text{ miglioramento medio} & \rightarrow \alpha = 0,0025 \end{cases}$
- $\begin{cases} \% \text{ istanza (valore base)} \geq \% \text{ miglioramento medio} & \rightarrow \alpha = 0,02 \\ \% \text{ istanza (valore base)} < \% \text{ miglioramento medio} & \rightarrow \alpha = 0,00125 \end{cases}$

In questo caso il processo di parametrizzazione non ha portato ad ulteriori miglioramenti rispetto al valore iniziale.

Nerf Barrier

Indica la percentuale di tempo in cui un cluster deve stare sopra la media per essere accettabile nell'iterazione successiva. È compreso tra 0 e 1.

Local search time limit

Massimo tempo per la fase di ricerca locale, che viene effettuata, se possibile, alla fine di ogni segmento. Il processo di ricerca locale trae vantaggio da tutti i vincoli euristici definiti dal nostro gruppo.

Max MIPS nodes to solve in feasibility check

Determina il massimo numero di nodi MIPS da far visitare al risolutore prima di abbandonare un feasibility check. Alcune soluzioni feasible potrebbero richiedere un numero di nodi MIPS superiori a quello imposto da questo parametro, ma siccome questo vincolo si può tradurre direttamente nell'impiego temporale per la verifica delle soluzioni, andrebbe tarato con cura anche sulla base dell'istanza da affrontare.

Heuristics scores

Ad ogni iterazione il punteggio assegnato alla coppia di euristiche applicate è aggiornato secondo i seguenti parametri:

- w_1 → punteggio assegnato all'euristica se la soluzione trovata è un ottimo globale;
- w_2 → punteggio assegnato all'euristica se la soluzione trovata è migliore della precedente;
- w_3 → punteggio assegnato all'euristica se la soluzione trovata è peggiore della precedente ma viene comunque accettata;
- w_4 → punteggio assegnato all'euristica se la soluzione trovata è peggiore della precedente ed è rifiutata.

Il punteggio assegnato all'euristica di rimozione è indipendente dal punteggio assegnato all'euristica di inserimento.

Reward for best segment heuristics

Punteggio applicato al peso dell'euristica migliore all'inizio del segmento, in base al risultato ottenuto nel segmento precedente.

Punishment for worst segment heuristics

Punteggio applicato al peso dell'euristica peggiore all'inizio del segmento, in base al risultato ottenuto nel segmento precedente.

9.8 Risultati complessivi

Il processo di parametrizzazione analitico è stato effettuato su sette parametri con un tempo computazionale di 10 minuti. L'ordine di analisi non è casuale, ma si è cercato di tarare innanzitutto i parametri più indipendenti e che non richiedevano uno studio comparato. Per i parametri soggetti a relazioni reciproche si sono cercati dei criteri di *tuning* in funzione dei risultati precedentemente ottenuti.

Come spiegato, l'ALNS è un algoritmo con grandi possibilità a fronte, però, di un numero elevato di parametri. La scelta di analizzare solo sette parametri non è, però, data solo da ragioni di efficienza, ma anche di efficacia. Come mostrato nel *Grafico 1* la parametrizzazione dei parametri di *Alpha* e di *Warmup Gamma* non ha portato alcun miglioramento; secondo noi ciò è dovuto prevalentemente a tre motivi:

1. Il processo di parametrizzazione preliminare è mostrato essere adatto in caso di generalizzazioni delle istanze (*Tabella 2*).

- Poiché l'ordine di analisi non è casuale e poiché, quindi, i valori dei parametri principali alla base del modello sono stati già assegnati, allora le variabili su cui si poteva agire per ottenere dei miglioramenti visibili sono già state fissate, vincolando fortemente la parametrizzazione.
- La terza ragione è collegata alla precedente. Ritenendo ragionevoli i criteri di individuazioni dei valori di *Alpha* e di *Warmup Gamma*, l'immobilità dell'algoritmo è dettata da caratteristiche intrinseche, non riconducibili alla parametrizzazione. Si ritiene, quindi, di poter essere arrivati al massimo sfruttamento del codice in esame.

Per tali motivi, si è deciso di non procedere ulteriormente con l'analisi di altri parametri, la cui influenza è pressoché assente per quanto riguarda il funzionamento complessivo dell'algoritmo. Da un miglioramento medio del 27% rispetto al benchmark si è ottenuto, mediante questo processo di parametrizzazione puntuale, un incremento al 58%.

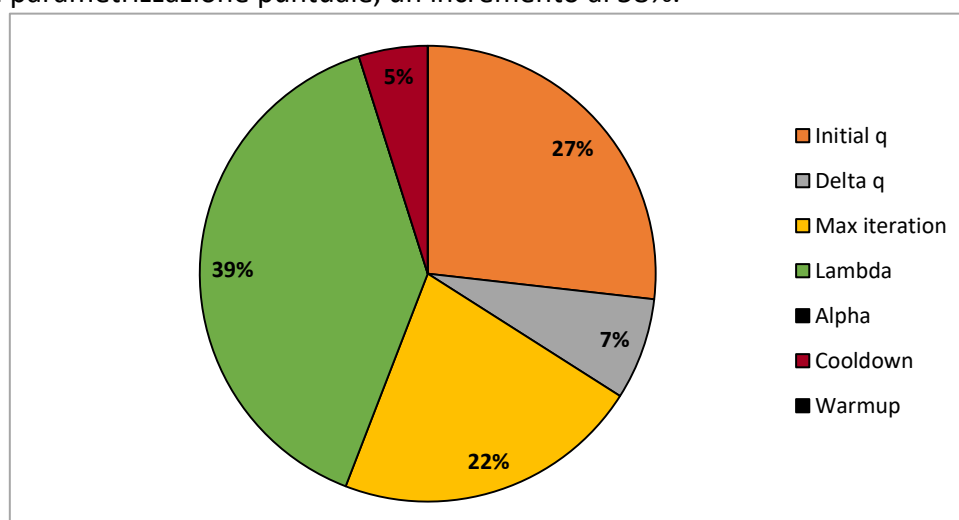


Grafico 1 – Impatto sul miglioramento medio rispetto al benchmark in base alla taratura di ciascun parametro

Dopo aver effettuato un'analisi puntuale su ogni specifica istanza, si è valutato globalmente l'impatto dei criteri adottati. Come mostrato dalla *Tabella 2*, in un'ottica di generalizzazione delle istanze, i valori individuati attraverso il processo di parametrizzazione preliminare sono i migliori in termini di miglioramento rispetto al benchmark.

		Delta q		Max Iteration w/o		Lamda	
		Base	36%	Base	38%	Base	46%
Initial q		Valore=1	12%	In funzione di w	27%	In funzione di Max iter	25%
Base	27%	in funzione di %rispettoc	14%	Valore =5	20%	In base al valore precedente	45%
10%	7%	Alpha		CooldownGamma		Warmup Gamma	
25%	-7%	Base	58%	Base	58%	Base	58%
50%	-14%	In funzione di Max iter	22%	In funzione di miglioramento%	36%	In funzione di miglioramento%	40%
		In base al valore precedente	15%	In funzione di miglioramento%	24%	In funzione di miglioramento%	21%

Tabella 2 – Miglioramenti medi in percentuale rispetto al benchmark in funzione dei parametri e dei relativi criteri di tuning

10. ANALISI

Una volta terminato il processo di parametrizzazione, ogni istanza è stata testata sui 30 minuti messi a disposizione dal problema.

Si è deciso di effettuare tre test per ogni istanza, poiché si è riscontrata una rilevante variabilità nei risultati forniti dall'algoritmo durante la fase di tuning dei parametri; variabilità dettata sia dall'applicazione di euristiche di natura randomica *Random Removal* e *Random Insertion* sia dall'esecuzione della *Local search*.

Per ogni istanza si è quindi calcolata la media e la deviazione standard dei tre valori ottenuti della funzione obiettivo. Come riportato dal *Grafico 2* una considerevole variabilità si focalizza solo in poche istanze, con una deviazione massima del 27% rispetto alla media.

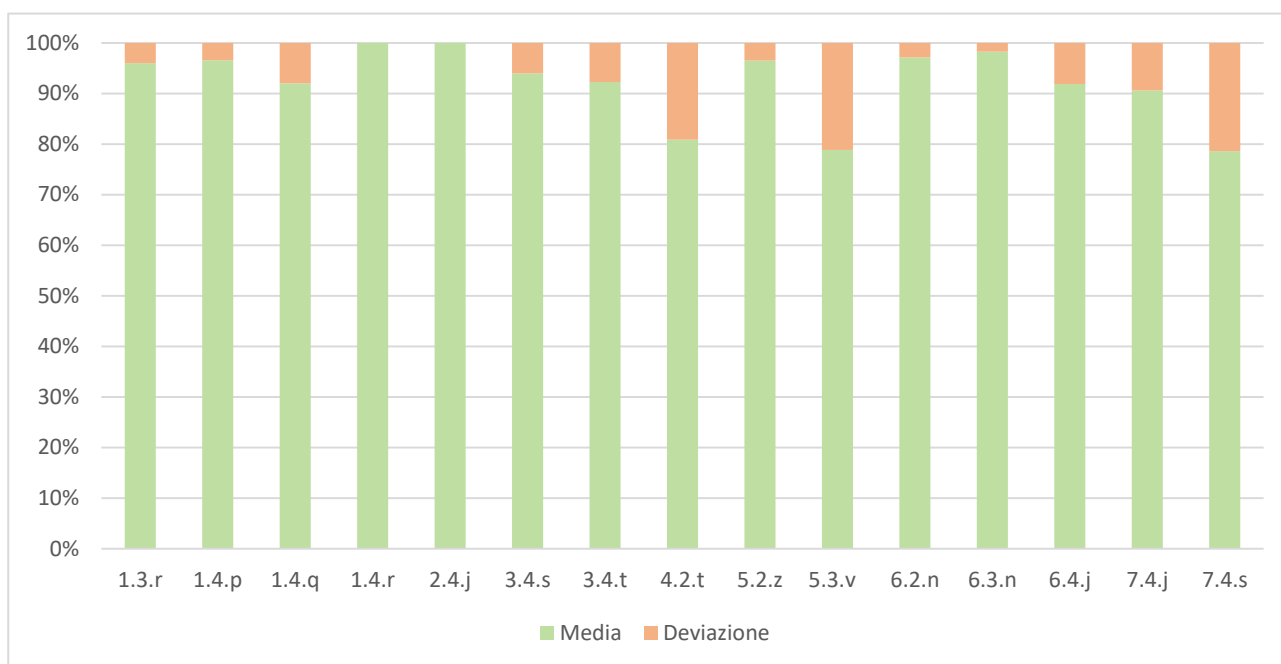


Grafico 2 - Rapporto tra deviazione standard e media

Da un'analisi più approfondita si è individuato un miglioramento medio, rispetto al benchmark, del + 62,27% considerando la soluzione migliore per ciascuna istanza, mentre l'*improvement* medio scende al + 46,76% nel caso in cui si considerino i valori medi.

In ogni caso due istanze non raggiungono mai il benchmark anche se lo scarto negativo è dell'8%, considerando i valori migliori, e del 12%, valutando i valori medi (percentuali già inglobate nel calcolo del miglioramento medio).

Queste performance fallimentari si sono verificate su istanze piccole (30 cluster) mentre non si sono riscontrate difficoltà nel caso in cui ci siano più cluster, denotando la capacità dell'algoritmo di gestire problemi complessi e di grandi dimensioni.

Per interpretare meglio i risultati di questo algoritmo si è effettuata un'analisi più puntuale sulla soluzione migliore per ciascuna istanza testata.

Come già delucidato precedentemente la nostra metaeuristica si struttura su tre livelli: costruttivo, ALNS e ricerca locale sulla migliore soluzione del segmento. Dapprima si è quindi deciso di calcolare l’impatto di ciascun passaggio sulla determinazione della soluzione finale.

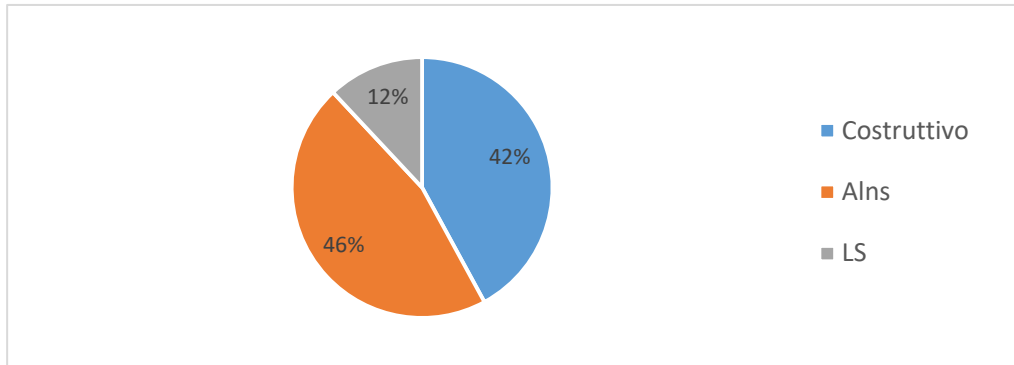


Grafico 3 –Miglioramento sulla funzione obiettivo ottenuti dai tre principali passi di cui si compone la nostra metaeuristica

Si può notare dal *Grafico 3*, come l’ALNS risulti essere fondamentale per il raggiungimento delle soluzioni ottenute. Allo stesso modo, contro ogni aspettativa, anche l’algoritmo costruttivo è un buon compromesso tra efficacia ed efficienza: garantisce il raggiungimento di soluzioni buone in tempi computazionali ridotti. L’influenza della *Local Search* è, invece, ragionevolmente limitata in funzione dello scopo per cui è stata implementata, ovvero intensificare la ricerca sulla base dei risultati ottenuti dall’ALNS.

Vista l’importanza rivestita dall’ALNS sono state approfondite le performance dell’algoritmo metaeuristico studiando le euristiche implementate e il loro utilizzo comparato.

Dai *Grafici 4 e 5* si può evincere un’apprezzabile omogeneità nella selezione delle euristiche di inserimento e rimozione da parte dell’algoritmo. In entrambi i casi gli operatori randomici hanno un’influenza più contenuta rispetto a quelli tradizionali. Sono state introdotte, inoltre, delle euristiche specifiche per il problema in esame, “*CloseToBarycenter*” e “*VehicleTime*”, le quali risultano essere due tra le più efficaci.

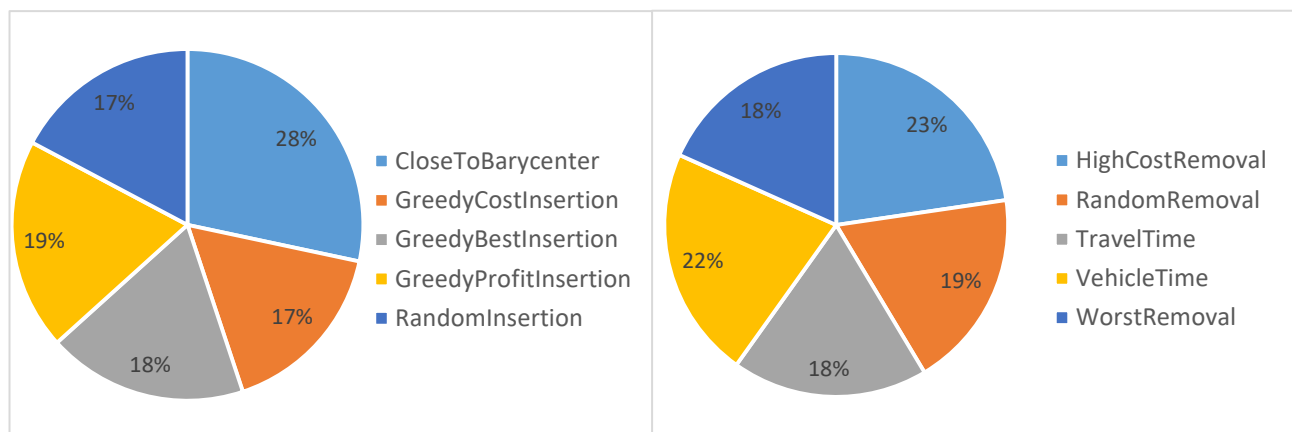


Grafico 4 – Percentuale media di estrazione dell’euristiche di distruzione (grafico a sinistra) e di riparazione (grafico a destra)

Analizzando il meccanismo di aggiornamento dei punteggi delle euristiche si riscontrano risultati simili a quelli dedotti dal grafico precedente: vi è una forte omogeneità dei pesi con una penalizzazione nei confronti degli operatori randomici, del “*WorstRemoval*” e del “*GreedyBestInsertion*” privilegiando, invece, le euristiche più conformate al problema.

Euristiche di Rimozione	Peso new	Euristiche di Inserimento	Peso New
<i>HighCostRemoval</i>	0,94	<i>CloseToBarycenter</i>	0,95
<i>RandomRemoval</i>	0,83	<i>GreedyCostInsertion</i>	0,83
<i>TravelTime</i>	0,86	<i>GreedyBestInsertion</i>	0,68
<i>VehicleTime</i>	0,86	<i>GreedyProfitInsertion</i>	0,91
<i>WorstRemoval</i>	0,81	<i>RandomInsertion</i>	0,81
Soluzioni peggiori accettate		42,96%	
Soluzioni peggiori non accettate		57,04%	

Tabella 3 – Media dei pesi ottenuti al termine del tempo computazionale e percentuali di soluzioni peggiorative accettate e non

Per quanto concerne la *Simulated Annealing*, si è verificato come l'obiettivo di diversificare la ricerca all'inizio di ogni segmento sia stato rispettato, determinando l'accettazione di circa 43% delle soluzioni peggiorative.

Sebbene i punteggi assegnati alle singole euristiche di inserimento siano indipendenti da quelli attribuiti alle euristiche di rimozione, si è deciso di valutare l'efficacia di ciascuna coppia estratta durante la ricerca.

Nell'eseguire questa analisi di tipo comparato, si sono selezionate le sole coppie di euristiche che hanno comportato un miglioramento della funzione obiettivo, per identificare gli abbinamenti più performanti.

Una buona combinazione risulta essere quella tra "*GreedyProfitInsertion*" e "*WorstRemoval*" grazie alla possibilità di considerare in modo congiunto il profitto e il costo (in termini di tempo) associati ad un cluster.

























	HighCostRemoval	RandomRemoval	TravelTime	VehicleTime	WorstRemoval
CloseToBarycenter	 3,7%	 6,4%	 8,1%	 4,0%	 2,7%
GreedyCostInsertion	 0,5%	-	 1,4%	 11,9%	 5,0%
GreedyBestInsertion	 1,3%	 5,0%	 2,1%	 2,4%	 4,0%
GreedyProfitInsertion	 2,8%	 3,7%	 2,0%	 9,1%	 10,2%
RandomInsertion	 2,4%	 2,8%	 1,2%	 5,3%	 2,1%

Tabella 4 – Miglioramento medio della funzione obiettivo ottenuto dall'estrazione di una determinata coppia di euristiche

BIBLIOGRAFIA

- [1] B. L. L. V. R. Golden, «The orienteering problem,» *Naval Research Logistics* , n. 34, pp. 307-318, 1987.
- [2] S. C. T. Butt, «A heuristic for the multiple tour maximum collection problem,» *Computers and Operations Research*, n. 21, p. 101–111, 1994.
- [3] M. G. B. W. E. Chao, «The team orienteering problem,» *European Journal of Operational Research*, n. 88, p. 474–474, 1996.
- [4] S. P. D. Røpke, «An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery,» *Transportation Science*, n. 40, pp. 455-472, 2006.
- [5] S. G. C. D. V. M. P. Kirkpatrick, «Optimization by Simulated Annealing,» *Science*, vol. 220, n. 4598, pp. 671-680, 1983.