Politecnico di Torino

Ingegneria Gestionale

Laurea Magistrale

Mateuristiche per il Fair Task Allocation Problem

Docente:
Prof. Marco Ghirardi

Candidata:
Marta Groia

Anno accademico 2019

Mateuristiche per il Fair Task Allocation Problem

Marta Groia

Dicembre 2019

Indice

Capitolo 1

Introduzione

Il seguente lavoro di tesi esamina la questione riguardante la distribuzione di compiti ad un insieme di agenti con la prerogativa di far fronte agli impegni contrattuali che li caratterizzano. I principali strumenti utilizzati per gestire il problema sono modelli di programmazione lineare. L'implementazione del problema è basata su un modello matematico noto come Fair Task Allocation Problem (FTAP). Mentre per il caso generale il problema è definito NP-hard, ciò significa che lo spazio delle soluzioni cresce in maniera esponenziale all'aumentare della dimensione del problema stesso, si dimostrerà come, per un determinato numero di compiti ed agenti, la sua risoluzione possa avere tempistiche polinomiali. L'obiettivo del lavoro di tesi riguarda dunque la pianificazione e lo sviluppo di un algoritmo che permetta l'allocazione ottima dei task in tempi di risoluzione accettabili. Per ottenere tale risultato si applicherà un algoritmo mateuristico. Il computer utilizzato per la risoluzione del modello matematico è un HP core i5-3317U, 1,7 GHZ, con 8 GB di memoria RAM, mentre i solver di ottimizzazione impiegati sono stati Xpress IVE e Cplex. La trattazione della tesi si articola in quattro capitoli. Il primo capitolo introduce in linea teorica gli elementi che sono stati adottati per lo sviluppo dello studio. Il secondo capitolo descrive il problema nel dettaglio con la definizione del modello matematico. Il terzo capitolo affronta la descrizione degli algoritmi realizzati per risolvere il problema. Si illustrano il funzionamento della soluzione approssimata, il procedimento metaeuristico implementato e come questo componente venga perfezionato con un miglioramento della soluzione. Il quarto capitolo riporta i risultati ottenuti.

1.1 Cos'è la Ricerca Operativa

Lo studio di questa tesi si basa su concetti di Ricerca Operativa (indicata con l'acronimo RO), materia che fornisce strumenti matematici di supporto alle attività decisionali in cui occorre gestire attività e risorse limitate al fine di massimizzare o minimizzare una funzione obiettivo.
Le prime applicazioni di RO risalgono alla metà del Settecento con G.Monge che esaminò un problema di trasporti e con F.Tylor che verso la fine dell'Ottocento elaborò un metodo di miglioramento dei processi produttivi.
Un ulteriore sviluppo di questa materia si ebbe poi durante la Seconda Guerra Mondiale con l'applicazione della RO in ambito militare per aumentare l'efficienza delle operazioni. A partire dal 1960 lo sviluppo della RO ha raggiunto i numerosi settori in cui la vediamo applicata oggigiorno.
Tra i problemi che possono essere risolti tramite la RO troviamo [2]:

- *finanza ed investimenti*:
  si tratta di definire quali investimenti sia più conveniente effettuare, quanto investire e quali siano i capitali necessari

- *pianificazione della produzione*:
  ha l'obiettivo di pianificare il livello ottimo di produzione e di utilizzazione delle risorse

- *localizzazione e dimensionamento degli impianti*:
  si tratta di problemi in cui si deve decidere dove installare impianti di produzione con lo scopo rifornire in modo ottimale aree distribuite su un territorio

- *project planning*:
  come progettare le molteplici attività di un progetto e le relative risorse coinvolte

- *telecomunicazione*:
  come dimensionare e collegare una rete di telecomunicazione in modo da garantire il traffico tra le varie origini e destinazioni e minimizzare il costo complessivo

- *determinazione dei turni del personale*:
  problema che si presenta ad esempio nella gestione del personale di un ospedale o nell'assegnazione dell'equipaggio di un volo

La Ricerca Operativa è quindi una metodologia che interessa svariate discipline, applicabile in molti contesti in cui è necessario un approccio modellistico che porta a migliorare l'efficacia delle decisioni da prendere. In linea

generale si possono individuare delle fasi ricorrenti nell'utilizzo della Ricerca Operativa come metologia di risoluzione che sono[3]:

1. studio del problema
2. raccolta dei dati
3. definizione del modello
4. implementazione soluzioni
5. analisi dei risultati

1.2 Modelli di Ottimizzazione Combinatoria

Tra i modelli matematici che ricoprono un ruolo centrale nella RO vi sono quelli che studiano Problemi di Ottimizzazione Combinatoria (COP) i quali trovano grande applicazione in numerose situazioni pratiche comuni. L'Ottimizzazione Combinatoria rappresenta una branca dell'Ottimizzazione Matematica collegata alla teoria degli algoritmi e alla ricerca operativa.
La caratteristica fondamentale di tali problemi è quella di avere insiemi ammissibili discreti, a differenza ad esempio della Programmazione lineare in cui l'insieme ammissibile delle soluzioni è continuo.

Ogni COP è caratterizzato da un certo grado di complessità che classifica il problema in base alla sua difficoltà di risoluzione. Le principali categorie di problemi sono definite come P e NP. La classe P comprende tutti quei problemi che possono essere risolti in un tempo polinomiale rispetto alla dimensione dell'istanza considerata. Questi problemi possono essere sempre risolti in modo efficiente ed indipendentemente dalle loro dimensioni. La seconda categoria invece è definita come «problemi di decisione le cui soluzioni possono essere verificate in tempo polinomiale, o la cui soluzione può essere trovata in tempo polinomiale con una macchina di Turing non deterministica»[4].

Alla classe NP appartengono anche le classi di problemi NP-ardui e NP-completi, che rappresentano delle versioni più difficili di un problema NP. La categoria di problemi NP-ardui può essere definita in linea generale come «la classe di problemi difficili almeno quanto i più difficili problemi di complessità P ed NP» [4]. Nonostante la classe NP-ardui contenga problemi potenzialmente risolvibili con un algoritmo dal tempo polinomiale, l'esistenza di algoritmi di tempo non polinomiale per il problema NP-arduo è sospettato, anche se non è ancora mai stato provato. Infine i problemi NP-completi possono essere definiti come «i problemi più difficili della classe NP» [4].

Perciò se si trovasse un algoritmo in grado di risolvere rapidamente un qualsiasi problema NP-completo, allora lo si potrebbe utilizzare per risolvere ogni problema NP.

1.3 Programmazione Lineare Misto-Intera

Nella programmazione matematica, un problema di ottimizzazione è inteso come un problema di Programmazione Intera (IP) se tutte le variabili sconosciute sono limitate a valori interi.
Andando più nello specifico, se il problema presenta una funzione obiettivo e vincoli lineari, si farà riferimento alla Programmazione Lineare Intera (ILP). Con il termine di Programmazione Lineare (LP) invece ci si riferisce a tutti quei problemi che presentano relazioni lineari tra funzione obiettivo e vincoli. IP e LP possono essere combinati insieme in problemi ILP. La Programmazione Lineare Intera è composta da elementi di LP che tengono conto delle relazioni lineari del modello e di elementi che restringono le variabili a numeri interi.

Comunemente problemi pratici ricadono in una variante di LP ancora diversa che prende il nome di Programmazione Misto-Intera (MIP) in cui le variabili non sono tutte obbligatoriamente intere. In generale un problema MIP può essere composto da variabili intere, continue e binarie. Come per la Programmazione Intera anche in questo caso se vincoli e funzione obiettivo sono legate da relazioni lineari si parlerà di Programmazione Lineare Misto-Intera (MILP).

1.4 MIP solvers

Attualmente i software realizzati specificamente per la risoluzione di questi problemi sono numerosi. Per questo lavoro di tesi sono stati considerati FICO *Xpress Optimizer* by FICO e IBM ILOG *Cplex 12.5* by IBM.
Entrambi i risolutori scelti implementano l'algorito risolutivo del Branch and Bound. Il BB fornisce un metodo di ricerca della soluzione ottima esplorando parzialmente l'insieme delle soluzioni ammissibili. In particolare si tratta di un metodo che enumera tutte le soluzioni del problema, considerando i seguenti elementi:

- operazione di branch: costruzione dell'albero delle soluzioni ammissibili
- disponibilità di una soluzione ammissibile

- operazione di bound: valutazione ottimistica della funzione obiettivo per le soluzioni rappresentate da ciascun nodo

L'algoritmo inizialmente partiziona (branch) un insieme S_0 delle soluzioni ammissibili del problema iniziale di minimo P_0 in sottoinsiemi S_1, S_2,..., via via più ristretti. Successivmente per ognuno di questi sottoinsiemi calcola un limite inferiore (lower bound) del valore ottimo della funzione obiettivo. Quei sottoinsiemi di soluzioni ammissibili per i quali il valore del limite non sia migliore del valore di funzione obiettivo di qualche soluzione ammissibile e nota x di P_0 sono esclusi da ulteriori partizioni. La procedura si arresta quando la migliore soluzione x, ha un valore di funzione obiettivo non peggiore del bound calcolato per ogni rimanente sottoinsieme.

1.5 Mateuristiche

Lo scopo di un problema di ottimizzazione è strettamente collegato alla definizione di una procedura di calcolo, ossia un *algoritmo*, in grado di risolvere in modo efficiente le istanze.
Il ragionamento che stà dietro alla risoluzione di un problema tramite l'applicazione di un algoritmo è quello di ottenere come output della procedura una soluzione ottima per una data istanza di intup.

Per la risoluzione di problemi polinomiali l'algoritmo che determina una soluzione ottima è definito *algoritmo esatto*. La ricerca della soluzione ottima tramite metodi esatti esplora l'intero insieme delle soluzioni tramite delle tecniche di enumerazione implicita come la programmazione dinamica, il branch-and-cut e il branch-and-bound.

Nell'affrontare problemi NP-ardui invece l'applicazione di un algoritmo esatto potrebbe risultare computazionalmente intrattabile, per questo motivo ci si trova nella necessità di ricorrere ad algoritmi che restituiscono una soluzione approssimata. Gli algoritmi che funzionano in questo senso sono le cosiddette *meta-euristiche* come ad esempio *tabu search* e *la ricerca locale*. Questi algoritmi possono essere applicati nella loro forma generale ad un'ampia gamma di problemi, tuttavia è spesso consigliabile adattarli alle caratteristiche del problema per ottenere un algoritmo più efficiente. A tale scopo sono state sviluppate nuove forme di algoritmi di natura ibrida in quanto uniscono metodi esatti e meta-euristici e prendono il nome di *mateuristiche*. Generalmente questi criteri si differenziano tra di loro poichè si adattano alle caratteristiche intriseche del problema in esame, ma possiedono poi una struttura di base fissa.

Le mateuristiche sono algoritmi di ottimizzazione che sfruttano la combinazione di programmazione matematica e metodi euristici per la risoluzione di problemi di ottimizzazione combinatoria. L'impiego di questi metodi ha permesso di migliorare la risoluzione di diversi problemi sia di interesse teorico sia del mondo reale. Generalmente la parte "base" dell'algoritmo è risolta da un risolutore di programmazione lineare MILP tramite un algoritmo euristico. Il risolutore MILP è affiancato da un algoritmo metaeuristico il cui scopo è quello di risolvere un sottoinsieme del problema generale. Di conseguenza essendo il sottoinsieme una porzione del problema intero si avrà una risoluzione più rapida.

I contributi forniti dalla ricerca in questo campo possono essere classificati sotto due aspetti: come i metodi metaeuristici possono migliorare i metodi esatti e come i metodi esatti possono migliorare gli algoritmi metaeuristici. Il secondo caso è quello più studiato per il fatto che potenziali benefici rispetto allo stato dell'arte sono più evidenti. In quest'area, si possono identificare due direzioni in cui la ricerca si sta orientando: la prima consiste nell'utilizzo di tecniche di programmazione matematica come componenti da includere in metodi metaeuristici, la seconda nel creare algoritmi metaeuristici derivati dalle logiche di funzionamento interno dei metodi di programmazione matematica [6].

1.5.1 Ricerca Locale

La Ricerca Locale è un metodo metaeuristico usato per risolvere problemi in cui è necessario trovare una soluzione tra un certo numero di candidati. Questo approccio si basa su un concetto molto semplice di risoluzione ossia quello di andare per tentativi.

Data la soluzione ammissibile di un problema s_1, con il relativo valore della funzione obiettivo $f(s_1)$. La Ricerca Locale in una prima fase definisce un intorno di s_1, poi ricerca al suo interno eventuali soluzioni migliori. Se, in questo "vicinato", si trova una soluzione s_2 per cui $f(s_2) < f(s_1)$, allora ci si sposta da s_1 a s_2 e si riparte da $s2$ con l'esplorazione del suo vicinato. Se invece nel vicinato di s_1 non si scopre nessuna soluzione migliore, vuol dire che s_1 è un minimo locale. Una volta arrivati al minimo locale, l'algoritmo si ferma e restituisce questo minimo come valore di output.

L'algoritmo di Ricerca Locale può essere riassunto in tre step [5]:

1. `Scegli una soluzione iniziale s1`

2. `Genera le soluzioni nel vicinato N(s1)`

3. Se in N(s1) c'è una soluzione s2 tale che f(s2) < f(s1), allora poni s1:=s2 e vai a 2, altrimenti STOP

Trattandosi di una Mateuristica non si ha la garanzia che la soluzione ottenuta sia quella ottima anzi tipicamente può essere anche molto distante dall'ottimo globale.

1.5.2 Tabu Search

Mentre nella Ricerca Locale l'informazione che viene memorizzata è quella della migliore soluzione corrente, l'algoritmo del Tabu Search ha invece l'intento di tenere memoria di alcune informazioni delle ultime soluzioni trovate. Applicando il concetto del Tabu search una volta trovato il minimo locale l'algoritmo prevede lo spostamento su una soluzione s_2 tale per cui è minimo il peggioramento della funzione obiettivo. A questo punto l'algoritmo esplora il vicinato della nuova soluzione e per evitare di ritornare su delle soluzioni già visitate utilizza le informazioni sulle ultime mosse effettuate. Più precisamente grazie alla memorizzazione delle ultime mosse è possibile proibire quelle mosse per evitare ricadute in punti già visitati.

L'algoritmo di Tabu Search si riassume nella seguente struttura [5]:

1. Scegli una soluzione iniziale s1, k:= 0 inizializzazione del contatore
2. Incrementa il contatore k e genera le soluzioni nel vicinato N(i;k)
3. Trova la soluzione s2 tale per cui f(s2) è minima con $s2 \in N$ e $s1 \neq s2$
4. Se f(s2) < f(s1), poni s1(ottima) := s2;
5. Aggiorna la tabu list inserendo la mossa che fa passare da s1 a s2 e poni s1 := s2
6. Se il criterio di arresto non è soddisfatto vai al passo 2, altrimenti STOP

Capitolo 2

Descrizione del problema e sviluppo del modello

2.1 Descrizione

Il problema affrontato in questo studio prende il nome di FTAP (Fair Task Allocation Problem) ed è descritto di seguito [1].

Nel mondo reale un'azienda manifatturiera impiega trasportatori a contratto per la spedizione dei loro prodotti. I volumi di trasporto sono combinati in anticipo a carichi di autocarri e i viaggi che ne derivano sono quindi assegnati a dei trasportatori per l'esecuzione. Poiché in molti paesi il settore logistico è dominato da specialisti dedicati a questa attività che offrono tariffe di trasporto molto competitive, è pratica comune per le compagnie di produzione esternalizzare questa mansione.

La strategia di assumere diversi partner per lo stesso tipo di lavoro è seguita principalmente per evitare le conseguenze negative derivanti dal dover fare affidamento su un'unica persona. Inoltre, un autotrasportatore potrebbe anche essere riluttante a firmare un contratto ad alto volume e dedicare significative frazioni della sua capacità ad un unico cliente.

Le motivazioni per cui le aziende cercano un supporto decisionale automatizzato nell'assegnazione dei lavori ai partner sono principalmente di due tipi. In primo luogo, con un processo automatizzato il processo di assegnazione può essere migliorato in termini di equità e di esecuzione del contratto, che potenzialmente porta a meno reclami da parte dei partner. In secondo luogo, assegnazioni manuali comportano il rischio di corruzione, dal momento che in genere ci sono alcuni aspetti positivi o sfavorevoli dei viaggi non coperti dai pagamenti previsti dal contratto, quindi i trasportatori possono essere tentati di rivolgersi agli spedizionieri per ottenere un servizio più conveniente. In particolare, il rischio di corruzione dovrebbe essere evitato per

quanto possibile nell'interesse dell'azienda focale di una catena di approvvigionamento. Questo è ancora più importante se si tratta di una catena di approvvigionamento globale che include partner in paesi meno sviluppati con un'alta suscettibilità alla corruzione.
Per comprendere meglio il problema si consideri l'esempio riportato in Figura 2.1.

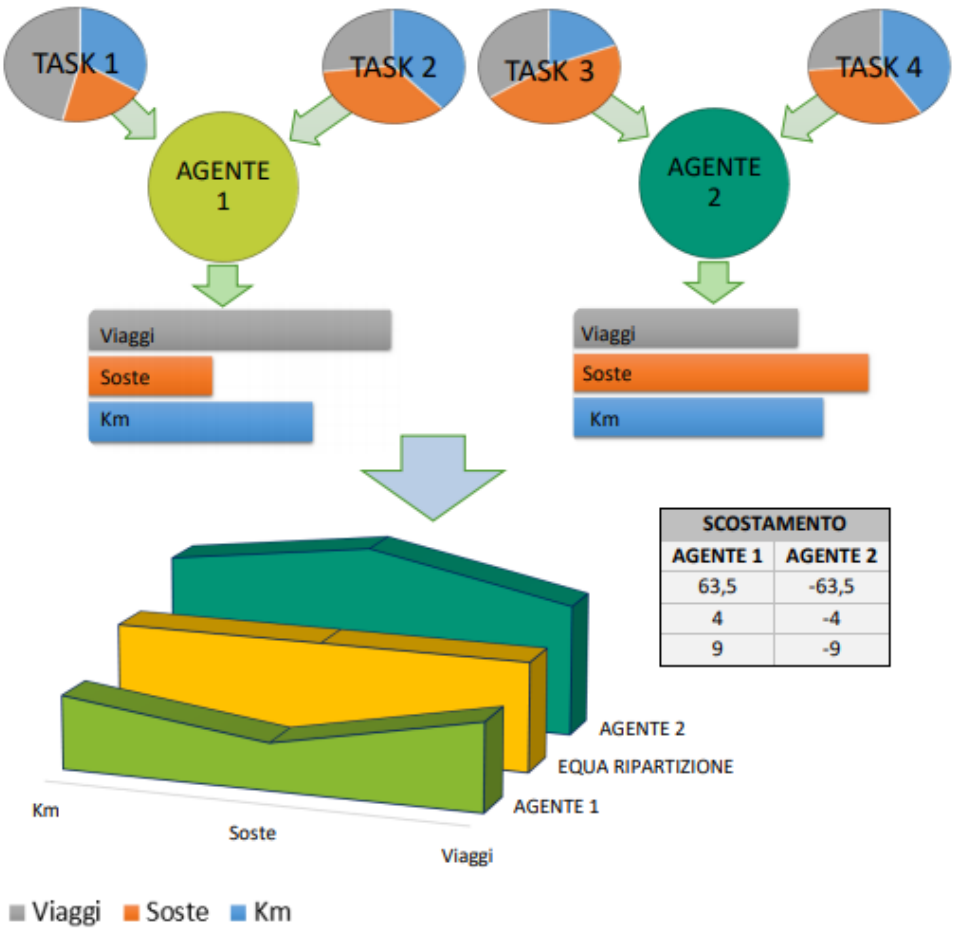


Figura 2.1: Diagramma esempio caso di Fair Task Allocation Problem

Il diagramma mostra quattro task differenti che devono essere ripartiti tra due agenti. I compiti da assegnare sono rappresentati da ellissi la cui colorazione rispecchia le diverse quote di dimensioni (viaggi, soste e km) da cui sono caratterizzati. Tramite un'allocazione casuale dei task agli agenti, all'agente uno sono affidati i task 1 e 2 mentre all'agente due i task 3 e 4, vengono a configurarsi i due diagrammi a barre, rappresentanti le diverse quote

di km, soste e viaggi affidate all'agente, che già ad una prima vista risultano essere sbilanciati. Il disequilibrio di questa allocazione è evidente nell'ultimo grafico in cui l'allocazione ottimale con equa ripartizione dei compiti è stata messa al confronto con l'allocazione casuale appena descritta.

Lo scostamento dalla soluzione ottimale è riportato in tabella in cui si hanno valori positivi quando all'agente vengono affidati compiti che non riempono la sua effettiva disponibilità, mentre valori negativi nel caso in cui l'agente sia sovracaricato.

In un contesto reale questa situazione comporterebbe insoddisfazione da parte degli agenti ma anche e sopratutto uno scorretto sfruttamento delle risorse disponobili da parte dell'azienda allocatrice dei task, per questo motivo risulta necessaria l'applicazione di un algoritmo che faciliti un'equa ripartizione dei compiti.

2.2 Campi di applicazione

Il contesto dei trasportatori non è l'unico a presentare il problema dell'equa ripartizione dei compiti che si ritrova infatti in diversi settori dove un gruppo di incarichi diversi deve essere assegnato a più lavoratori. In queste circostanze l'assegnazione di turni o compiti per i lavoratori può portare a insoddisfazione se fatto in modo ingiusto.

Uno dei primi e più importanti campi di applicazione in cui è stata considerato il problema dell'equa ripartizione è quello dell'allocazione della larghezza di banda nelle reti di telecomunicazione. In quest'area sono utilizzati flussi continui con coppie di origine-destinazione predefinite, che portano ad algoritmi i quali aumentano simultaneamente i flussi su tutti i percorsi fino a quando i collegamenti non sono saturi. In questo campo è stato definito il cosidetto algoritmo di riempimento progressivo, che viene considerato come uno dei concetti standard di equità all'interno delle telecomunicazioni o delle applicazioni di rete.

L'allocazione delle risorse è un altro campo in cui l'equità gioca un ruolo importante. Un esempio di vincolo che potrebbe emergere in questo caso è che prima o poi ogni compito sarà in grado di utilizzare la risorsa richiesta. Un requisito più rigoroso è rappresentato dall'equità proporzionale, in cui la differenza delle risorse assegnate ai diversi compiti non sarà mai maggiore di uno, assicurando che tutti i compiti abbiano un accesso simile alle risorse.

Concretamente, il problema si pone ad esempio negli aeroporti più precisamente nel turnover dei lavoratori a terra. In genere, questi lavoratori sono qualificati per una varietà di compiti diversi, che vanno da quelli di attività interne, ad esempio alla cintura bagagli, alle attività all'aperto, come

il carico e lo scarico degli aerei o la manutenzione di strade e piste. I vari tipi di compiti devono essere coperti da turni che vengono poi assunti dai lavoratori durante le ore di lavoro. A parte le proprietà di base di un turno, come il giorno della settimana, il suo totale ore, e il pagamento, questo può avere altre caratteristiche rilevanti ai fini della correttezza, come ad esempio l'attrattiva del compito, lo sforzo fisico associato o l'esposizione al rumore o a condizioni meteorologiche estreme. Le quotazioni target dei lavoratori possono variare a causa dei diversi contratti o desideri personali.

2.3 Definizione del modello

In termini generali, il problema considerato consiste nel dover distribuire n lavori a m agenti che dovrebbero eseguirli. Ogni lavoro è caratterizzato da delle proprietà numeriche e ogni agente ha valori obiettivo che vorrebbe raggiungere per ogni dimensione. Per essere più precisi, l'obiettivo del problema è quello di ridurre al minimo la somma delle deviazioni tra: i valori obiettivo degli agenti e la somma delle proprietà numeriche dei lavori assegnati moltiplicata per un peso dipendente dalle dimensioni su tutti gli agenti e le dimensioni.

Consideriamo T come un insieme di compiti con $|T|= n$ e A come un insieme di agenti con $|A|= m$ che dovrebbero eseguirli. Definiamo un insieme di dimensioni identificate da D.
Ogni compito t ha proprietà numerica $p_{t,d} \in N$ in ogni dimensione d e ogni agente ha valori obiettivo $u_{a,d} \in R$ in tutte le dimensioni, che dovrebbero essere soddisfatti il più fedelmente possibile dagli incarichi. Inoltre, introduciamo pesi positivi w_d per far si che le deviazioni delle dimensioni siano paragonabili tra di loro.
L'obiettivo del problema di ottimizzazione è quello di minimizzare la somma delle deviazioni assolute tra assegnazione e target per ogni agente in ogni dimensione, dove ogni compito deve essere assegnato esattamente ad un agente.

2.3.1 Fair Task Allocation Problem

- Le variabili decisionali sono:
  - $y_{t,a}$: variabile binaria che denota i task t assegnati all'agente a ($y_{t,a} = 1$) oopure no ($y_{t,a} = 0$).
  - $\delta_{a,d}$: deviazione assoluta dell'obiettivo dell'agente in una data dimensione.

- I parametri del modello sono:
  - $T = \{1, 2, ..., n\}$ è l'insieme dei task che devono essere eseguiti dagli agenti.
  - $A = \{1, 2, ..., m\}$ è l'insieme degli agenti che eseguono i task.
  - $D = \{1, 2, ..., l\}$ è l'insieme delle dimensioni prese in considerazione.
  - $p_{t,d}$: proprietà numerica che caratterizza ogni task in ciascuna dimensione.
  - $u_{a,d}$: valore target che ha un agente in relazione ad una dimensione.
  - w_d: peso positivo che rende le deviazioni nelle diverse dimensioni comparabili tra di loro.

2.3.2 Modello

$$O.F. = min \sum_{d \in D} \sum_{a \in A} w_d \cdot \delta_{a,d}$$

Subject to:

$$\sum_{a \in A} y_{t,a} = 1 \qquad \forall t \in T \quad (2.1)$$

$$\delta_{a,d} \geq \sum_{t \in T} y_{t,a} \cdot p_{t,d} - u_{a,d} \qquad \forall a \in A;\ d \in D \quad (2.2)$$

$$\delta_{a,d} \geq u_{a,d} - \sum_{t \in T} y_{t,a} \cdot p_{t,d} \qquad \forall a \in A;\ d \in D \quad (2.3)$$

$$y_{t,a} \in \{0, 1\} \qquad \forall a \in A;\ t \in T \quad (2.4)$$

$$\delta_{a,d} \in R \qquad \forall a \in A;\ d \in D \quad (2.5)$$

2.3.3 Interpretazione dei vincoli

Riferimento vincolo	Interpretazione
2.1	Assicura che ogni compito sia assegnato esattamente ad un solo agente
2.2	Assicura che le deviazioni $\delta_{a,d}$ siano sempre positive. Il vincolo assume l'ugualianza nel caso in cui l'agente a riceve troppo nella dimensione d
2.3	Speculare al vincolo 2.2. Il vincolo assume l'ugualianza nel caso in cui l'agente a riceve troppo poco nella dimensione d.
2.4	Definisce il dominio della variabile $y_{t,a}$ come binaria.
2.5	Definisce il dominio della variabile $\delta_{a,d}$ come l'insieme dei numeri reali R

Tabella 2.1: Descrizione dei vincoli

2.3.4 Generazione delle istanze

Vista la mole di dati relativi ad un'istanza del problema (numero di agenti, numero di task, descrizione del set di dimensioni come: chilometri percorsi, numero di fermate e numero di viaggi effettuati), è stato necessario creare una cartella di file dati apposito per ogni caso su cui applicare il modello. Sono stati considerati sei differenti casi con 75, 100 e 250 tasks che devono essere assegnati a 5 ed 8 agenti. Per oguno di questi sei casi di studio sono state generate 5 istanze random tramite l'utilizzo di comandi mosel, riportati in appendice A. Per la generazione delle istanze sono stati considerati i seguenti intervalli dei parametri riguardanti le dimensioni D del problema:

- *chilometri percorsi* da 11 a 253 km;
- *numero di fermate* da 1 a 4 fermate;
- *numero di viaggi* da 1 a 40 viaggi.

Ad esempio, un file di dati del tipo "75-5dataset1.txt" contiene i dati relativi ad un'istanza da 75 tasks e 5 agenti. Il file di dati viene letto dal programma in fase di inizializzazione in modo da acquisire tutte le costanti necessarie. Per un esempio effettivo di file di istanza, vedere l'appendice C.

Considerando una ripartizione equa dei compiti, in ogni dimensione il valore target degli agenti $u_{a,d}$, con numero di agenti pari a 5 ed 8, è stato

fissato rispettivamente al 20% e 12.5% del valore di $p_{t,d}$. Generalizzando il parametro è descritto dalla Formula 2.6.

$$u_{a,d} = \frac{1}{A} \sum_{t \in T} p_{t,d} \qquad \forall a \in A;\ d \in D \quad (2.6)$$

Infine è stato definito il parametro w_d, come da Formula 2.7, che permette di confrontare tra di loro i valori delle diverse dimensioni.

$$w_d = \frac{1000}{\sum_{t \in T} p_{t,d}} \qquad \forall d \in D \quad (2.7)$$

Capitolo 3

Implementazione algoritmi

Per l'implementazione del modello e la sua risoluzione sono stati utilizzati il software di ottimizzazione FICO@Xpress e Cplex, che utilizzano l'approccio Branch and Bound per risolvere problemi di programmazione lineare. Il tutto combinato con metodi euristici per trovare rapidamente buone soluzioni.

Per l'avviamento del programma è stato utilizzato il prompt dei comandi che permette di avere una risoluzione del problema più rapida. In Figura 3.2 è riportato il codice per il lancio di Xpress tramite Cmd mentre in Figura 3.1 quello per Cplex.

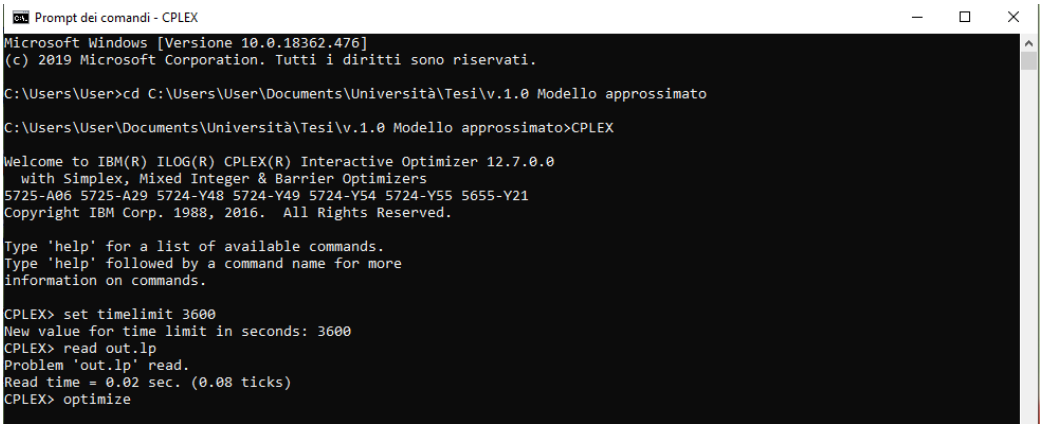


Figura 3.1: Prompt dei comandi: lancio di Cplex

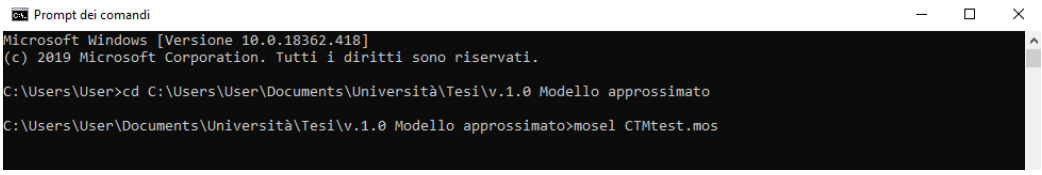


Figura 3.2: Prompt dei comandi: lancio di Xpress

I comandi utilizzati possono essere descritti come segue:

- `cd (nome della cartella)`: comando usato per spostarsi nella cartella che contiene il file da risolvere
- `mosel (nome del file)`: comando che permette la risoluzione del programma scritto su Xpress
- `cplex(...)`: comando che permette la risoluzione del programma scritto su Cplex
- `set time limit(...)`: comando che imposta il limite massimo di tempo per la risoluzione del problema
- `optimize(...)`: comando che esegue l'ottimizzazione del programma

3.1 Algoritmi

A partire dal modello matematico descritto al Paragrafo 2.3.2 sono state implementate tre diverse strategie. La prima è un modello approssimato, in quanto il tempo computazionale con un modello esatto risultava essere per alcune istanze troppo elevato ed è stato perciò necessario inserire un limite di tempo. Il secondo modello invece applica un approccio euristico che permette di ottenere una soluzione in tempi rapidi ma non ottimale. Infine il terzo ed ultimo approccio cosiste nel miglioramento della soluzione ottenuta nella fase precedente con implementazione della strategia multi-start.

3.1.1 Algoritmo approssimato

La risoluzione del problema tramite un algoritmo approssimato è apparsa inevitabile dal momento che per alcune istanze l'utilizzo di un algoritmo esatto era estremamente dispendioso.
Questo tipo risoluzione permette di introdurre un limite di tempo, nel caso di studio impostato alla soglia di 3600 secondi, oltre il quale il solver si ferma e restituisce come FO l'ultima soluzione trovata. Per esprimere questo vincolo di tempo su Xpress è stato utilizzato il comando `setparam("XPRS MAXTIME",TL)` dove TL rappresenta il tempo limite. Tramite questo approccio si avanno perciò soluzioni ottenute entro il limite di tempo che sono da considerarsi come soluzioni ottime e soluzioni approssimate le quali invece sono state troncate perchè superavano l'ora come tempo di calcolo, si veda Figura B.1 in appendice A.

In questa fase di risoluzione del problema è stato inoltre effettuato un confronto dei dati ottenuti tramite l'utilizzo di diversi solver Xpress e Cplex, che hanno portato in alcuni casi a risultati distanti in termini di tempi computazionali. Tale operazione di confronto è stata implementata grazie all'utilizzo del comando `exportprob(EP MIN, "out.lp", FO)` che permette di esportare il problema scritto in Xpress su un file di lettura per Cplex.
Il listato del programma in Xpress IVE è riportato in appendice A.

3.1.2 Mateuristica

Avendo riscontrato, con la risoluzione di un algoritmo approssimato, la presenza di istanze che non raggiungono l'ottimo nell'intervallo di un'ora, si è scelto di applicare un algoritmo euristico che permettese di ottenere soluzioni in tempi più rapidi. L'euristica applicata ha lo scopo di ridurre il tempo di calcolo del solver nei casi in cui sia stato troppo lento. Questo obiettivo si raggiunge tramite il rilassamento del vincolo di binarietà della variabile y, che nel nuovo modello viene fissata come minore o uguale ad 1.
L'euristica si comporta nel seguente modo:

1. Le variabili y che dopo una prima iterazione del modello assumono valori pari ad 1 o 0 vengono fissate a quei valori

2. Tra le variabili continue viene ricercato il valore massimo che sarà fissato ad 1

3. Il problema viene reiterato fino a quando tutte le variabili y assumono valori pari ad 1 o 0

Questo algoritmo non garantisce una soluzione ottima ma permette di ottnere delle soluzioni accettabili anche per quei casi in cui l'algoritmo approssimato non raggiungeva l'ottimo.

L'applicazione dell'euristica non è stata ritenuta necessaria per i casi di istanze aventi 8 agenti in quanto con il metodo approssimato si raggiunge una soluzione ottima con dei buoni tempi di calcolo. Per quanto riguarda i casi con 5 agenti sono state invece selezionate due istanze per ogni caso che con il metodo approssimato riportavano un runtime maggiore rispetto alle altre. Il programma in linguaggio Mosel è riprodotto in appendice A.

3.1.3 Miglioramento della soluzione

Per attenuare gli svantaggi causati dai due precedenti approcci quali runtime elevati per i metodi esatti e soluzioni potenzialmente troppo lontane dal

valore di ottimo per la mateuristica, è stato progettato un terzo algoritmo che mettesse d'accordo i primi due. Si tratta del criterio "Miglioramento della soluzione".

Dal punto di vista della programmazione, è un'estensione del codice scritto per la mateuristica, si veda il listato riportato in appendice A.

L'obiettivo è stato quello di applicare dinamicamente i vincoli binari solamente a un sottoinsieme delle variabili decisionali y in modo da migliorare iterativamente la soluzione. In altre parole, partendo dalla soluzione dell'istanza mateuristica, una porzione di variabili decisionali è stata rilassata e ottimizzata ottenendo quindi una soluzione che per definizione fosse un LB della mateuristica e un UB del metodo esatto.

Nell'implementazione dell'algoritmo la dimensione dei sottoinsiemi di variabili da non riottimizzare è stata tenuta fissa al valore di 10. Per quanto riguarda il criterio di STOP dell'algoritmo invece è stato assunto come riferimento un numero massimo di *non miglioramenti* della soluzione corrente rispetto alla migliore trovata.

3.1.4 Multi-start

La strategia Multi-start rappresenta un elemento appropriato per sviluppare gli algoritmi di risoluzione di quei problemi in cui risulta più efficiente trovare una soluzione piuttosto che applicare una procedura di ricerca locale. I metodi Multi-start sono caratterizzati da due fasi: la prima consiste nella generazione della soluzione la seconda fase invece nel miglioramento della stessa. Nel caso di studio l'implementazione di questo metodo ha permesso di ridurre l'influenza, che ha sulla soluzione finale, la generazione random dei sottoinsiemi selezionati dalla mateuristica.
In appendice A è riportato il listato con il codice in linguaggio Mosel.

Capitolo 4

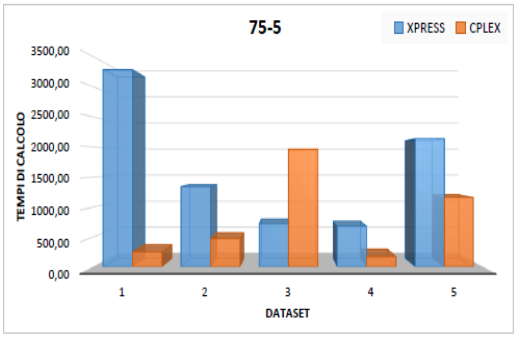
Risultati

In questo capitolo sono riportati e confrontati i risultati ottenuti dai diversi lanci dei solver, con l'obiettivo di definire un algoritmo che permetta di ottenere una soluzione che perda il meno possibile in termini di qulità e venga trovata in tempi accettabili. La prima sezione mostra i risultati ottenuti dal confronto dei due diversi solver utilizzati con il metodo approssimato. Successivamente sono riportati i dati ottenuti con l'utilizzo dell'algoritmo euristico, del miglioramento della soluzione e del metodo multi-start.

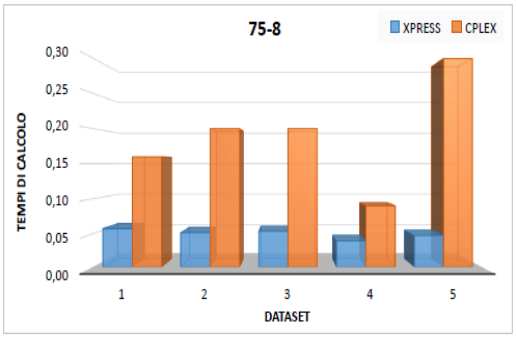
4.0.1 Xpress vs Cplex

I dati ottenuti da questa prima analisi dell'algoritmo sono riportati nell'appendice B e hanno permesso di effetuare le considerazioni di seguito riportate.

Dal confronto dei due differenti solver utilizzati per la risoluzione si può notare come Xpress sia più rapido nei problemi con 8 agenti Figura 4.1b, Figura 4.2b e Figura 4.3b mentre Cplex risulti più rapido nei casi con 5 agenti Figura 4.1a, Figura 4.2a e Figura 4.3a.

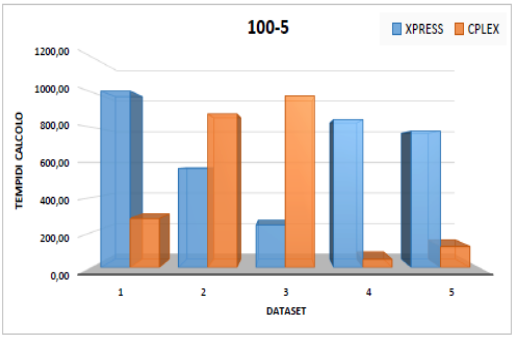


(a) 5 agenti

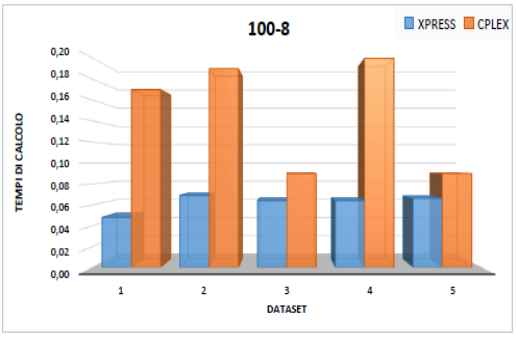


(b) 8 agenti

Figura 4.1: Confronto dei solver con 75 tasks

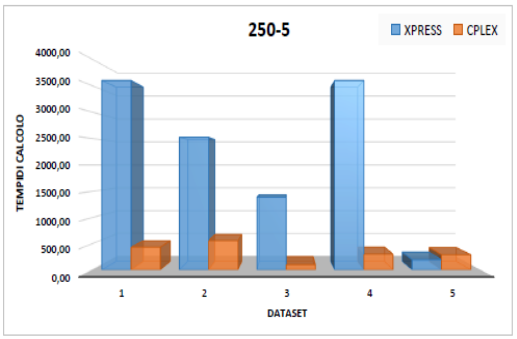


(a) 5 agenti

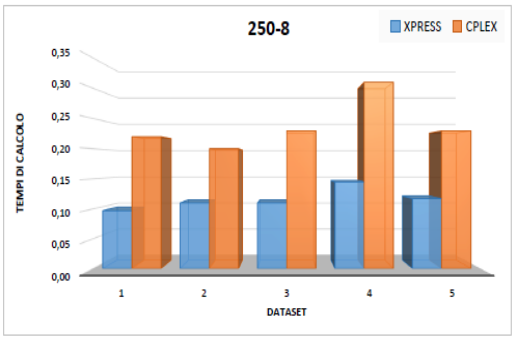


(b) 8 agenti

Figura 4.2: Confronto dei solver con 100 tasks



(a) 5 agenti

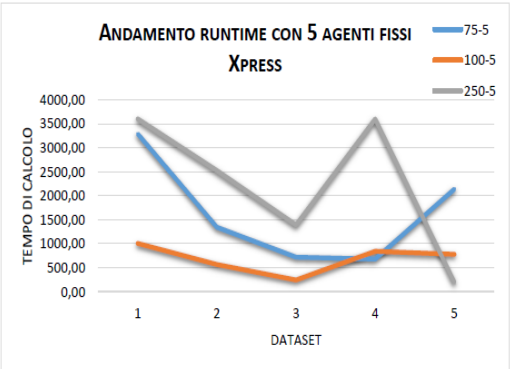


(b) 8 agenti

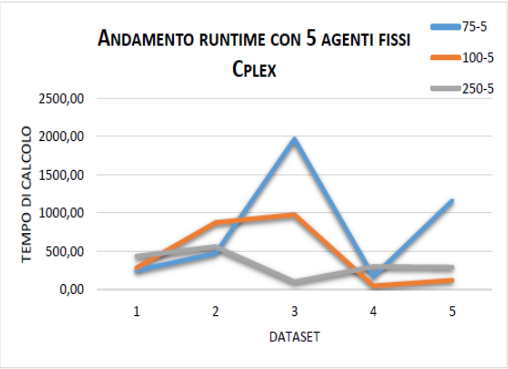
Figura 4.3: Confronto dei solver con 250 tasks

L'aumento degli agenti da 5 ad 8 ha reso i run time molto inferiori per entrambi i solver con valori che restano sotto il secondo per ogni test. Questo risultato potrebbe sembrare anomalo dal mometo che all'aumento delle istanze ci si aspetta una maggiore complessità computazionale, ma trattandosi di un problema NP-arduo è possibile che l'andamento della funzione tempo presenti delle oscillazioni e non sia monotono.

Nella Figura 4.5 e Figura 4.5 è stato analizzato l'andamento del tempo di calcolo del programma all'aumentare dei tasks con numero di agenti fissato. Dai risultati ottenuti si può notare che nel caso di risoluzione con Xpress il runtime riporti un andamento di crescita all'aumentare del numero di compiti Figura 4.4a e Figura 4.5a. Al contrario nel caso di utilizzo del solver Cplex sembra che tale valore non sia influenzato dall'aumento del numero di tasks Figura 4.4b e Figura 4.5b.

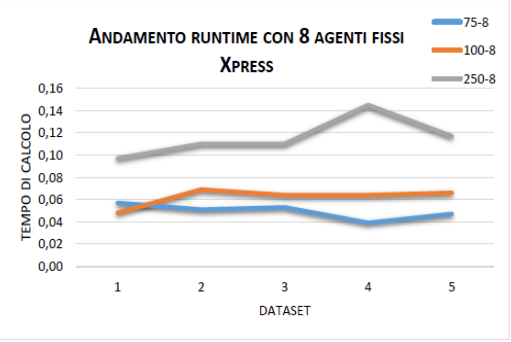


(a) Xpress

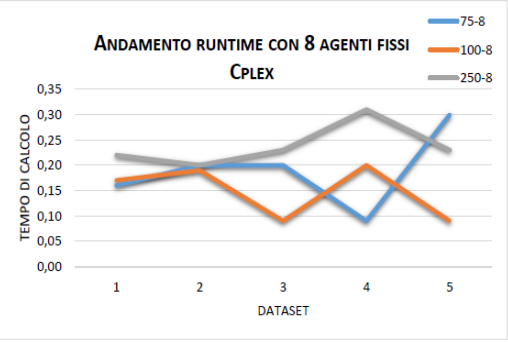


(b) Cplex

Figura 4.4: Andamento runtime con 5 agenti



(a) Xpress



(b) Cplex

Figura 4.5: Andamento runtime con 8 agenti

Per quanto riguarda la risoluzione del problema entro il tempo limite si presentano due casi in cui Xpress non è riuscito ad ottenere la soluzione ottima entro l'ora. Si presentano però altri casi in cui il tempo di risoluzione risulta essere oneroso e per questo saranno trattati con l'applicazione della Mateuristica.

4.0.2 Mateuristica vs Modello approssimato

L'applicazione della mateuristica al modello iniziale ha restituito i dati riportati nella Tabella 4.1 i quali sono stati poi confrontati con le soluzioni del modello approssimato Tabella 4.2.

Caso	Dataset	FO	Runtime [s]
75-5	1	105,035	0,361
	5	84,331	0,512
100-5	1	63,022	0,281
	4	71,827	0,38
250-5	1	18,186	0,302
	4	29,187	0,389

Tabella 4.1: Soluzioni con mateuristica

Caso	Dataset	FO	Runtime [s]
75-5	1	18,365	3287,224
	5	5,629	2137,043
100-5	1	13,238	1005,34
	4	17,907	843,078
250-5	1	4,981	3600,021
	4	1,818	3599,577

Tabella 4.2: Soluzioni con modello approssimato

Dai risultati ottenuti è evidente come la Mateuristica riesca a ridurre notevolmente i tempi di calcolo del solver che sono passati dall'ordine dell'ora a quello del mezzo secondo vedi Figura 4.6.

Per quanto riguarda la FO si nota che i valori, con l'introduzione dell'euristica sono aumentati e quindi peggiorati. Questo risultato era prevedibile dal momento che l'approccio utilizzato permette di ottenere una soluzione

approssimata e non ottima. Dalla Figura 4.7 è ben visibile il gap che intercorre tra la soluzione del metodo approssimato e quella ottenuta con la Mateuristica.

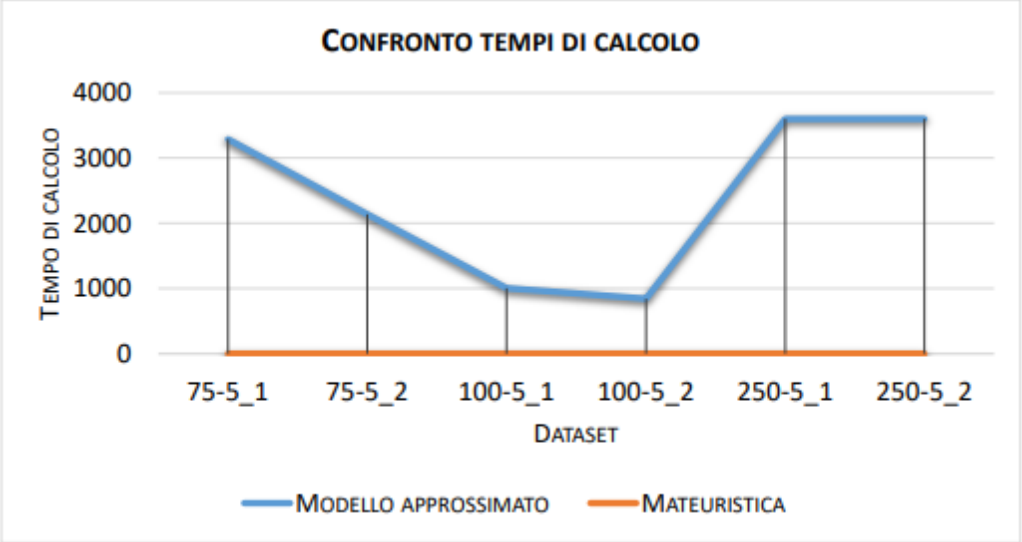


Figura 4.6: Comparazione tempo di calcolo Modello approssimato e Mateuristica

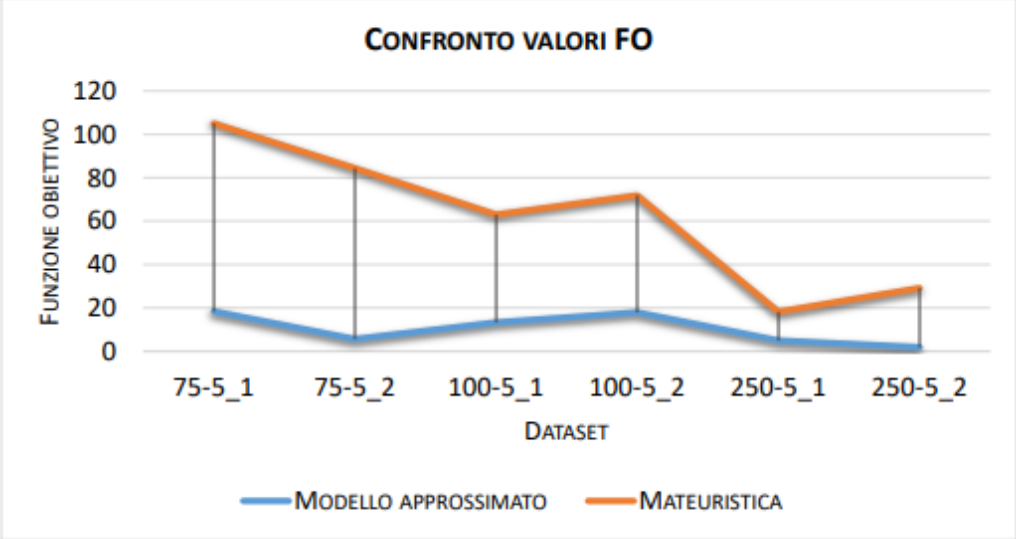


Figura 4.7: Comparazione FO Modello approssimato e Mateuristica

4.0.3 Miglioramenti della Mateuristica

Per l'implementazione dell'algoritmo di miglioramento della soluzione è stato necessario, nella fase iniziale, definire la dimensione di due prametri: l'ampiezza del sottoinsieme di variabili da non riottimizzare e il numero massimo di *no-miglioramenti* che restituisce il programma.
Come riportato nel capitolo 3.1.3 la dimensione del sottoinsieme è stata impostata a 10 variabili mentre per il criterio di stop sono stati effettuati dei test di confronto per individuare quale fosse il valore di *no-miglioramenti* più adatto al problema. Il solver è stato quindi lanciato per tre differenti valori del parametro *no- miglioramenti* che sono: 2, 3 e 10. Inoltre avendo impostato una generazione dei sottoinsiemi random a seconda dell'intervallo le soluzioni possono avere valori anche molto distanti, per questo motivo si è pensato di calcolare la media delle soluzioni trovate con 4 lanci del solver.
I risultati ottenuti sono riportati in appendice B. Per quanto riguarda il valore della funzione obiettivo è evidente come questo migliori all'aumentare del parametro. In Figura 4.8 si nota come per ogni differente istanza l'istogramma con *no-miglioramenti* minore di 10 risulti sempre inferiore rispetto agli altri due casi. In Figura4.9 si nota invece come il tempo di calcolo risulti maggiore nei casi in cui lo stop dell'algoritmo è impostato con valori di *no-miglioramenti* maggiori. Alla luce di tali risultati la scelta del parametro è ricaduta su *no-miglioramenti* < 3 che è risultato essere un ottimo compromesso.

Figura 4.8: Andamento FO al variare del parametro no-improvement

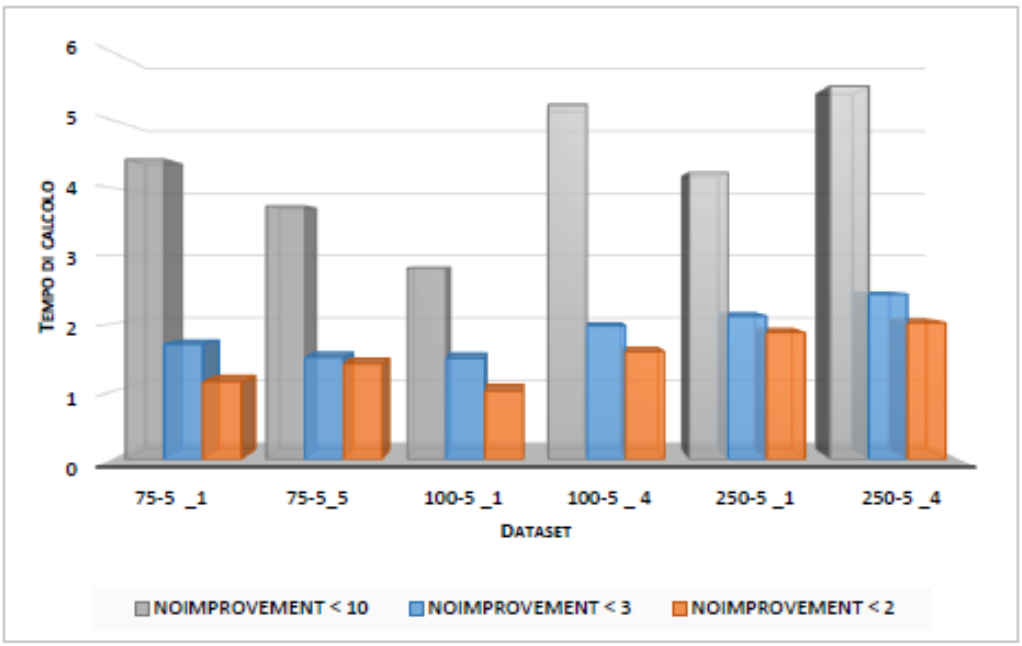


Figura 4.9: Andamento tempo di calcolo al variare del parametro no-improvement

I dati definitivi ottenuti dall'implementazione dell'algoritmo di miglioramento della soluzione con *sottoinsieme = 10* e *no-miglioramenti < 3* sono riportati in Tabella 4.3.

Caso	Dataset	FO	Runtime [s]
75-5	1	35,232	1,693
	5	26,539	1,500
100-5	1	20,977	1,483
	4	29,423	1,96
250-5	1	11,130	2,112
	4	7,550	2,431

Tabella 4.3: Soluzioni con estensione mateuristica

Dai risultati ottenuti si nota come l'algoritmo di miglioramento della soluzione ha ridotto enormemente il gap che si presentava tra la soluzione ottenuta con l'euristica e quella del modello semplice Figura 4.10.

Anche per quanto riguarda i tempi di calcolo si riscontrano dei buoni risultati, in quanto questi non si sono discostati di molto da quelli trovati con l'applicazione della Mateuristica e restano quindi molto bassi 4.11.

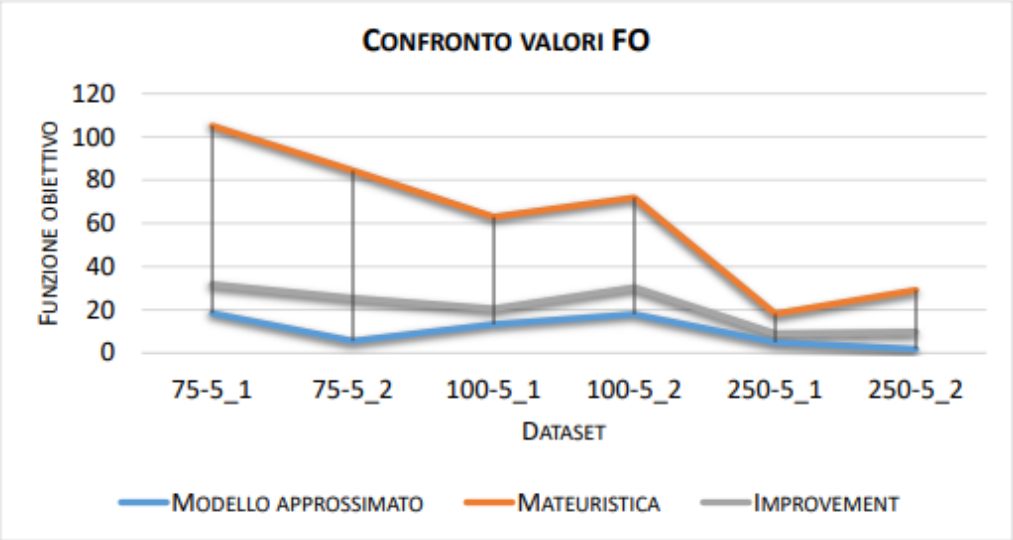


Figura 4.10: Comparazione FO dei tre modelli utilizzati

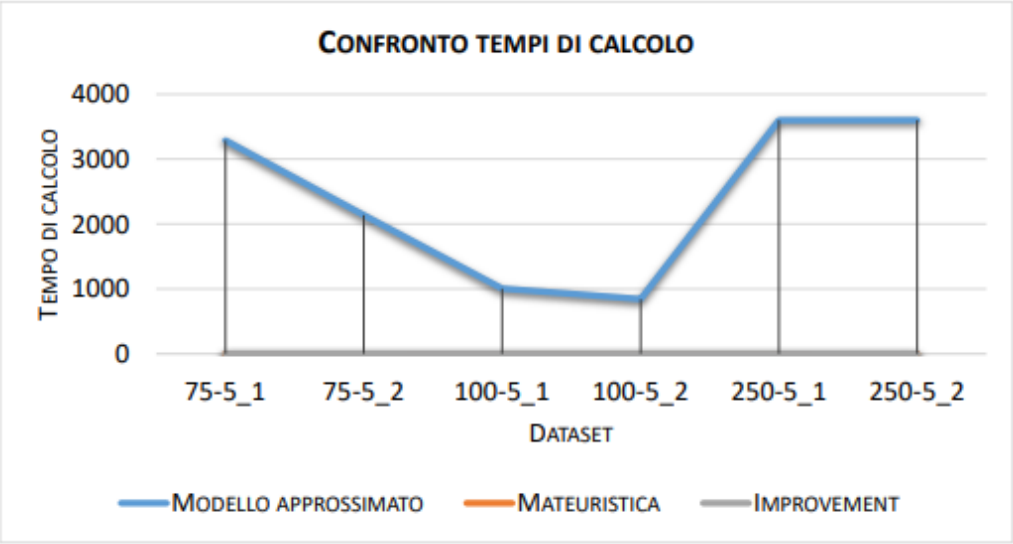


Figura 4.11: Comparazione tempo di calcolo dei tre modelli utilizzati

La fase finale di miglioramento della Mateuristica ha previsto l'applicazione di un algoritmo Multi-start che permettesse di eliminare l'effetto della generazione randomica di sottoinsiemi presente nella procedura di miglioramento della soluzione. Sono stati registrati i dati ottenuti con il lancio del solver Xpress con 10, 20 e 30 ripetizioni dell'algoritmo. I dati ottenuti sono riportati in appendice B e hanno permesso di identificare come migliore soluzione la reiterazione dell'algoritmo per 10 volte in quanto si ottengono buoni risultati della FO e in tempi preferibili rispetto che con un numero maggiore di iterazioni si veda Figura 4.12 e Figura 4.13.

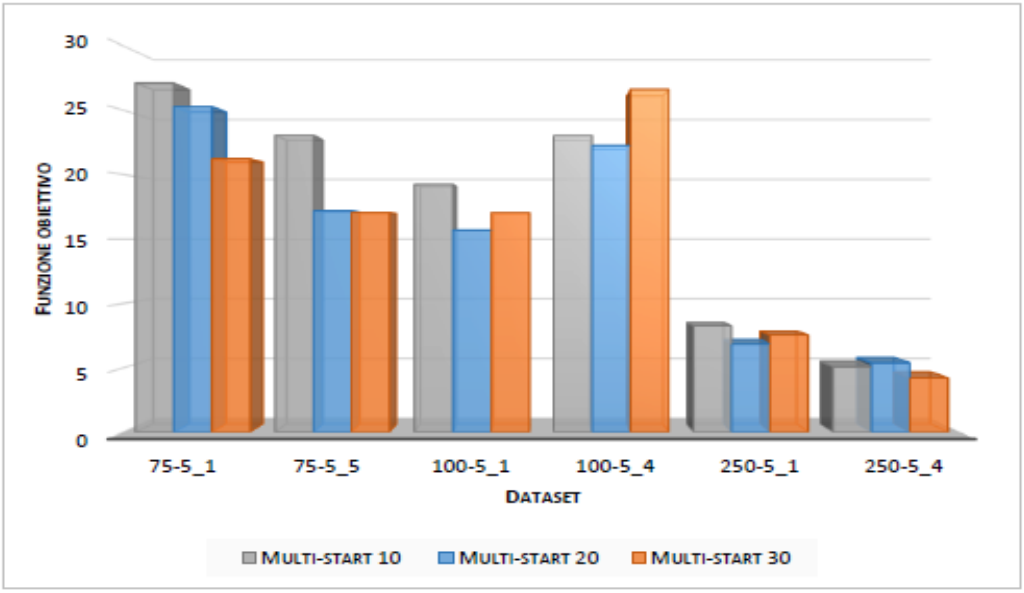


Figura 4.12: Confronto FO con differenti multi-start

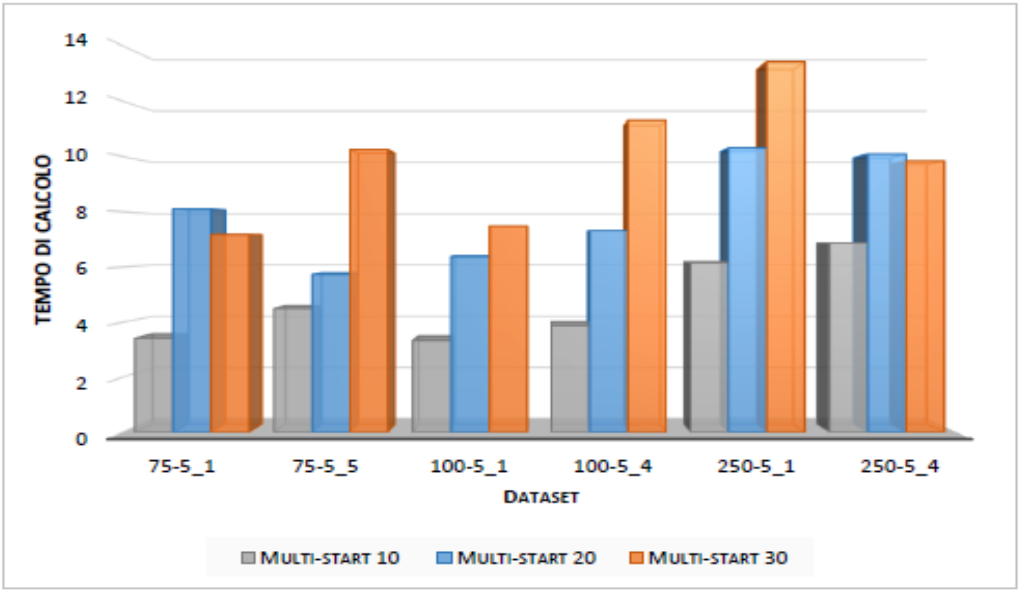


Figura 4.13: Confronto tempo di calcolo con differenti multi-start

L'applicazione del Multi-start con 10 iterazioni ha restituito i risultati in Tabella 4.4 che sono stati confrontati con i risultati dei metodi precedentemente applicati.

Caso	Dataset	FO	Runtime [s]
75-5	1	27,108	3,419
	5	23,044	4,493
100-5	1	19,236	3,34
	4	23,023	3,88
250-5	1	8,240	6,16
	4	5,085	6,866

Tabella 4.4: Soluzioni con algoritmo multi-start

In Figura 4.14 si nota come il valore della funzione obiettivo si sia avvicinato ulteriormente alla curva dei valori ottimi restituiti dal modello approssimato. Considerando i tempi di calcolo Figura 4.15 la soluzione è trovata in tempi molto più bassi rispetto all'utilizzo del modello approssimato, anche se rispetto all'implementazione di mateuristica e miglioramento della soluzione Figura 4.16 l'algoritmo multi-start risulta essere più lento.

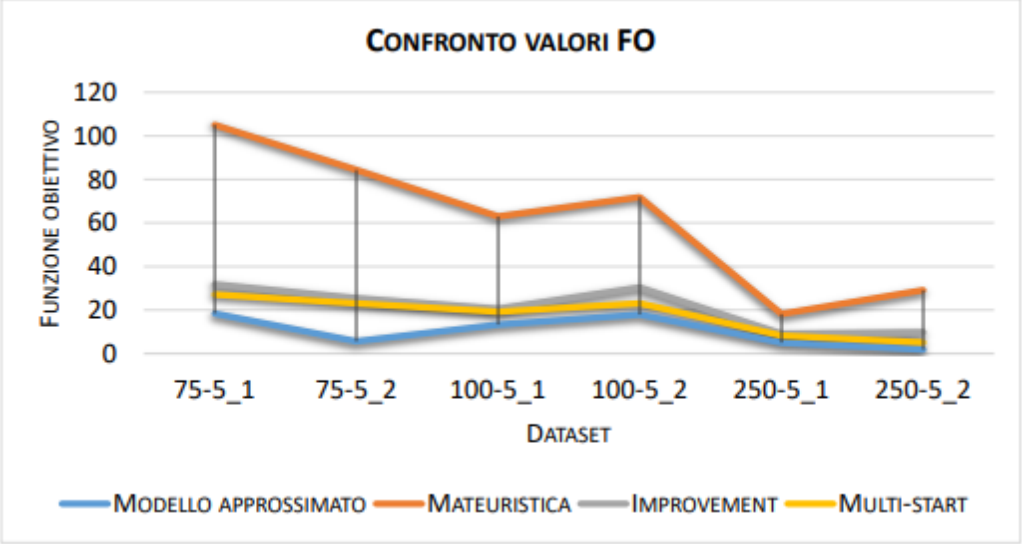


Figura 4.14: Comparazione FO dei modelli utilizzati

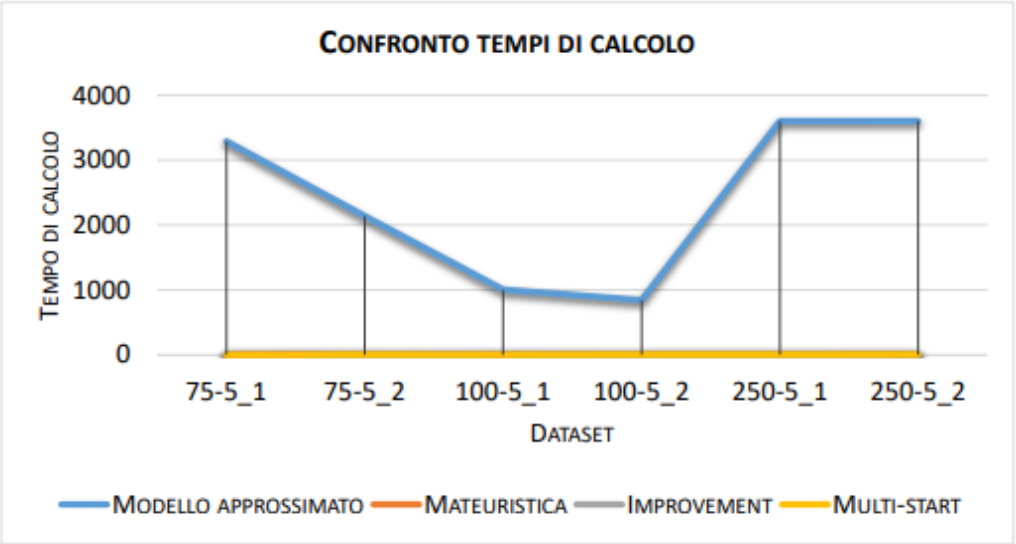


Figura 4.15: Comparazione tempo di calcolo dei modelli utilizzati

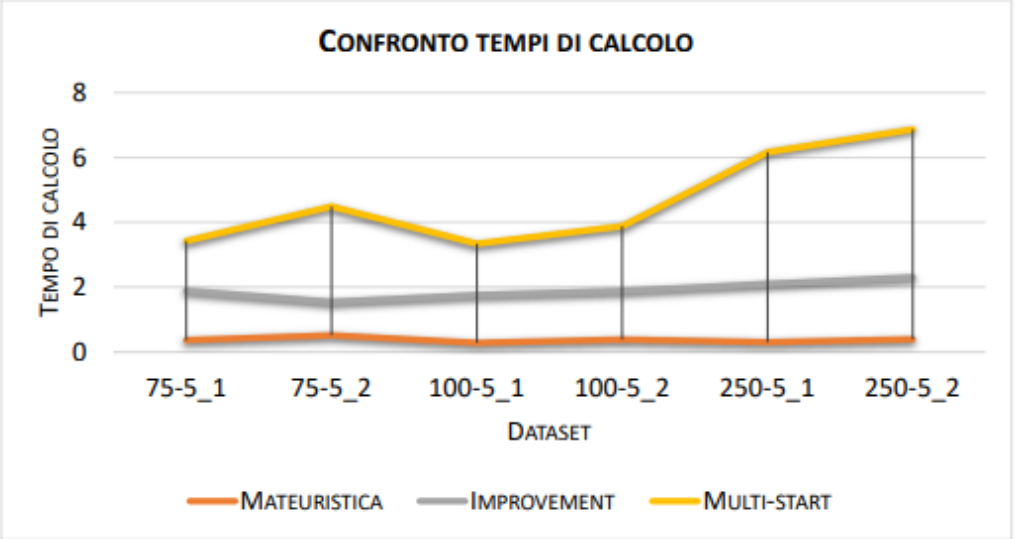


Figura 4.16: Comparazione tempo di calcolo mateuristica, miglioramento soluzione e multi-start

Capitolo 5

Conclusioni

L'obiettivo del lavoro di tesi riguarda la progettazione e lo sviluppo di un algoritmo che permetta l'allocazione equa di compiti ad un gruppo di agenti, in tempi di risoluzione accettabili.

Il raggiungimento di tale scopo è stato ottenuto tramite l'applicazione di un algoritmo mateuristico affiancato ad un metodo di miglioramento della soluzione con approccio multi-start.

L'utilizzo di un approccio approssimato ha consentito di individuare quali fossero le istanze che richiedevano l'implementazione di un'euristica per la loro risoluzione ed ha inoltre permesso un confronto dei risultati ottenuti con due diversi solver di ottimizzazione da cui è emersa una risoluzione più rapida con l'utilizzo di Cplex.

I risultati ottenuti con l'implementazione della mateuristica invece, hanno riportato sul modello notevoli miglioramenti in termini di tempi di calcolo, mentre l'algoritmo di miglioramento della soluzione multi-start ha permesso una riduzione del gap della funzione obiettivo rispetto al suo valore ottimale.

I riscontri ottenuti nelle diverse fasi di test permettono di confermare la competitività dell'algoritmo sviluppato rispetto ad altri modelli di risoluzion già presenti come il Tabu Search che richiedono modellizzazioni più complesse.

Appendice A

Listato del programma in Xpress IVE

Si riportano tutti i codici utilizzati per la generazione delle istanze e quelli dei diversi modelli di risoluzione del problema che sono stati lanciati.

```
model Generazione_istanze
uses "mmxprs"
uses "mmsystem"

setparam("XPRS_verbose", true)
setparam("REALFMT", "%1.15f")

declarations
    NAGENTI = 8
    NTASK =250
    NDIMENSIONI = 3
    NTIPI=6
    MinKM = 11
    MaxKM = 253
    MinVIAGGI = 1
    MaxVIAGGI = 40
    MinSOSTE = 1
    MaxSOSTE = 4

    a= 1..NAGENTI
    t= 1..NTASK
    d= 1..NDIMENSIONI

    Agenti: array(a) of integer
    Task: array(t) of integer
    Dimensioni: array(d) of string
    Proprieta: array (t,d) of integer
    Obiettivo: array (a,d) of integer
    w: array (d) of real
    y: array(t,a) of mpvar
    s: array (a,d) of mpvar

end-declarations

!codice usato per generare le istanze random
writeln("VETTORE AGENTI")
forall(i in a)
    write(" (",i,") ", i  )
    writeln("")

writeln("VETTORE TASK")
forall(i in t)
    write(" (",i,") ", i  )
    writeln("")

writeln("MATRICE PROPRIETA")
forall(i in t) do
    forall(j in d) do
        if (j=1) then
            randomvalue := MinKM +round(random*(MaxKM+1-MinKM) - 0.5)
        end-if
        if (j=2) then
            randomvalue := MinVIAGGI +round(random*(MaxVIAGGI+1-MinVIAGGI) -
0.5)
        end-if
        if (j=3) then
            randomvalue := MinSOSTE +round(random*(MaxSOSTE+1-MinSOSTE) - 0.5)
        end-if
    write("(",i," ",j,") ", randomvalue," ")
    end-do
writeln(" ")
end-do
```

```
model Modello approssimato
uses "mmxprs"
uses "mmsystem"

setparam("XPRS_verbose", true)
setparam("REALFMT", "%1.15f")

declarations    ! variabili da impostare+decisionali
    MinKM = 11
    MaxKM = 253
    MinVIAGGI = 1
    MaxVIAGGI = 40
    MinSOSTE = 1
    MaxSOSTE = 4
    MinTIPO= 1
    MaxTIPO= 100

    a: set of integer
    t: set of integer
    d: set of integer

    Agenti: array(a) of integer
    Task: array(t) of integer
    Dimensioni: array(d) of string
    Proprieta: array (t,d) of integer
    Obiettivo: array (a,d) of integer
    w: array (d) of real
    Tipologia: array (t,h) of integer
    Capacita: array (a,h) of integer
    y: array(t,a) of mpvar
    s: array (a,d) of mpvar


end-declarations

initializations from "75-5dataset1.txt"   ! è il file che contiene i dati

    Agenti
    Task
    Dimensioni
    Proprieta

end-initializations

finalize(a)
finalize(t)
finalize(d)

forall(i in a, j in d) do
    Obiettivo(i,j) := integer(0.2 * sum(k in t) Proprieta(k,j))
end-do

forall(i in d) do
    w(i) := 1000 / sum(q in t) Proprieta(q,i)
end-do

FO:= sum(i in a, j in d) w(j) * s(i,j)

forall (i in t ) do         !vincolo (2.1)
    sum( j in a) y(i,j)=1
end-do
```

```
forall (i in a, j in d ) do           !vincolo (2.2)
        s(i,j)>= (sum(k in t)( y(k,i) * Proprieta(k,j)))-Obiettivo(i,j)
end-do

forall (i in a, j in d) do           !vincolo (2.3)
    s(i,j)>= Obiettivo(i,j)-(sum(k in t)( y(k,i) * Proprieta(k,j)))

end-do

forall (i in a, j in t ) do  !vincolo (2.4)
    y(j,i)is_binary
end-do

setparam("XPRS_MAXTIME", 3600)       !Comando che inserisce il limite di tempo
exportprob(EP_MIN, "out.lp", FO)    !Comando per l'esportazione del programma su
un file testo di lettura per Cplex


start_time:=time(SYS_NOW)
minimize(FO)
end_time:= time(SYS_NOW)
writeln(" run time= ",end_time-start_time," ms ")
writeln(" la FO vale ", getobjval )

end-model
```

```
model Mateuristica
uses "mmxprs"
uses "mmsystem"

setparam("XPRS_verbose", true)
setparam("REALFMT", "%1.15f")

declarations
    MinKM = 11
    MaxKM = 253
    MinVIAGGI = 1
    MaxVIAGGI = 40
    MinSOSTE = 1
    MaxSOSTE = 4

    a: set of integer
    t: set of integer
    d: set of integer

    Agenti: array(a) of integer
    Task: array(t) of integer
    Dimensioni: array(d) of string
    Proprieta: array (t,d) of integer
    Obiettivo: array (a,d) of integer
    w: array (d) of real
    y: array(t,a) of mpvar
    s: array (a,d) of mpvar

end-declarations

initializations from "75-5dataset1.txt"   ! è il file che contiene i dati

    Agenti
    Task
    Dimensioni
    Proprieta

end-initializations

finalize(a)
finalize(t)
finalize(d)

forall(i in a, j in d) do
    Obiettivo(i,j) := integer(0.2 * sum(k in t) Proprieta(k,j))
end-do

forall(i in d) do
    w(i) := 1000 / sum(q in t) Proprieta(q,i)
end-do

FO:= sum(i in a, j in d) w(j) * s(i,j)

forall (i in t ) do          !vincolo (2.1)
    sum( j in a) y(i,j)=1
end-do

forall (i in a, j in d ) do           !vincolo (2.2)
        s(i,j)>= (sum(k in t)( y(k,i) * Proprieta(k,j)))-Obiettivo(i,j)
end-do

forall (i in a, j in d) do          !vincolo (2.3)
```

```
    s(i,j)>= Obiettivo(i,j)-(sum(k in t)( y(k,i) * Proprieta(k,j)))

end-do

forall (i in a, j in t ) do  !vincolo (2.4)
    y(j,i)<= 1               !rilassamento del vincolo di binarieta'
end-do

setparam("XPRS_MAXTIME", 3600)
exportprob(EP_MIN, "out.lp", FO)
start_time:=time(SYS_NOW)

!Mateuristica
flag:=1
while(flag=1)do
    flag:=0
    minimize(FO)
    tmax:=0
    amax:=0
    currentmax:= 0.0

    forall(i in t, j in a)do

        if(getsol(y(i,j))=0)then                          !codice che fissa a 0 e
1 le variabili che dopo il primo minimize sono gia' a 0 o 1
            y(i,j)=0
        end-if
        if(getsol(y(i,j))=1)then
            y(i,j)=1
        end-if

        if(getsol(y(i,j))>0 and getsol(y(i,j))<1)then   !codice che cerca le
variabili continue con valore massimo per fissarle ad 1
        flag:=1
            if(currentmax<getsol(y(i,j))) then
                currentmax:=getsol(y(i,j))
                tmax:=i
                amax:=j
            end-if
        end-if
    end-do
    if (flag=1)then
        y(tmax,amax)=1
    end-if
end-do


end_time:= time(SYS_NOW)
writeln(" run time= ",end_time-start_time," ms ")
writeln(" la FO vale ", getobjval )

end-model
```

```
model Miglioramento_soluzione
uses "mmxprs"
uses "mmsystem"

!setparam("XPRS_verbose", true)
setparam("REALFMT", "%1.15f")

declarations
    MinKM = 11
    MaxKM = 253
    MinVIAGGI = 1
    MaxVIAGGI = 40
    MinSOSTE = 1
    MaxSOSTE = 4

    a: set of integer
    t: set of integer
    d: set of integer

    Agenti: array(a) of integer
    Task: array(t) of integer
    Dimensioni: array(d) of string
    Proprieta: array (t,d) of integer
    Obiettivo: array (a,d) of integer
    w: array (d) of real
    y: array(t,a) of mpvar
    s: array (a,d) of mpvar

end-declarations

initializations from "75-5dataset1.txt"   ! è il file che contiene i dati

    Agenti
    Task
    Dimensioni
    Proprieta

end-initializations

finalize(a)
finalize(t)
finalize(d)

forall(i in a, j in d) do
    Obiettivo(i,j) := integer(0.2 * sum(k in t) Proprieta(k,j))
end-do

forall(i in d) do
    w(i) := 1000 / sum(q in t) Proprieta(q,i)
end-do

FO:= sum(i in a, j in d) w(j) * s(i,j)

forall (i in t ) do          !vincolo (2.1)
    sum( j in a) y(i,j)=1
end-do

forall (i in a, j in d ) do           !vincolo (2.2)
        s(i,j)>= (sum(k in t)( y(k,i) * Proprieta(k,j)))-Obiettivo(i,j)
end-do

forall (i in a, j in d) do           !vincolo (2.3)
```

```
        s(i,j)>= Obiettivo(i,j)-(sum(k in t)( y(k,i) * Proprieta(k,j)))

end-do

forall (i in a, j in t ) do  !vincolo (2.4)
        y(j,i) <=1
end-do


forall(i in t, j in a) do
    yto0(i,j) := y(i,j) = 0
    yto1(i,j) := y(i,j) = 1
    sethidden(yto0(i,j),true)
    sethidden(yto1(i,j),true)
end-do

setparam("XPRS_MAXTIME", 3600)
exportprob(EP_MIN, "out.lp", FO)

start_time:=time(SYS_NOW)
flag:=1
while(flag=1)do
    flag:=0
    minimize(FO)
    tmax:=0
    amax:=0
    currentmax:= 0.0

    forall(i in t, j in a)do
        if(getsol(y(i,j))=0)then
            sethidden(yto0(i,j), false)
        end-if
        if(getsol(y(i,j))=1)then
            sethidden(yto1(i,j), false)
        end-if
    end-do

    forall(i in t, j in a) do
        if(getsol(y(i,j))>0 and getsol(y(i,j))<1)then
            flag:=1
            if(currentmax<getsol(y(i,j))) then
                currentmax:=getsol(y(i,j))
                tmax:=i
                amax:=j
            end-if
        end-if
    end-do
    if (flag=1)then
            sethidden(yto1(tmax,amax), false)
    end-if
end-do

! Integer problem

forall(i in t, j in a) do
    y(i,j) is_binary
end-do

minimize(FO)
current_of := getobjval;
noimprovements := 0
writeln("La soluzione prima del while vale=", getobjval)
```

```
! Improvement

taskdim:=getsize(t)
while(noimprovements < 3) do                    !criterio di STOP: mi fermo dopo
3 cicli che non portano a miglioramenti della soluzione
    f := integer(round((taskdim*random)+0.5))   !generazione random del
sottoinsieme di variabili
    if(f<=taskdim-10)then
        fminimo:=f
        fmassimo:=fminimo+10
    end-if
    if(f>=taskdim-10)then
        fmassimo:=f
        fminimo:=fmassimo-10
    end-if

    forall(i in t, j in a) do
        if (getsol(y(i,j)) = 0) then
            sethidden(yto0(i,j), false)
            sethidden(yto1(i,j), true)
        end-if
        if (getsol(y(i,j)) = 1) then
            sethidden(yto0(i,j), true)
            sethidden(yto1(i,j), false)
        end-if
    end-do

    forall(i in t, j in a | i>fminimo and i<fmassimo ) do
        sethidden(yto0(i,j),true)
        sethidden(yto1(i,j),true)
    end-do


    minimize(FO)

    writeln(".....................................")
    writeln("La soluzione vale=", getobjval)

    if (getobjval < current_of) then
        writeln("La soluzione piu' recente e' minore quindi e' migliorata e
vale=", getobjval )
        current_of := getobjval
        noimprovements := 0
        writeln("quindi sovrascrivo questa soluzione", getobjval, "e il numero
di insuccessi viene azzerato infatti e' =", noimprovements)
    else
    noimprovements := noimprovements + 1;
    writeln("La soluzione piu' recente e' peggiorata, quindi rimane quella
precedente che vale=",current_of)
    writeln( "è aumentato il numero di non miglioramenti= ", noimprovements)
    end-if

end-do

end_time:= time(SYS_NOW)

writeln(" run time= ",end_time-start_time," ms ")
writeln(" la FO vale ", getobjval )

end-model
```

```
model Multistart
uses "mmxprs"
uses "mmsystem"

!setparam("XPRS_verbose", true)
setparam("REALFMT", "%1.15f")

declarations
    MinKM = 11
    MaxKM = 253
    MinVIAGGI = 1
    MaxVIAGGI = 40
    MinSOSTE = 1
    MaxSOSTE = 4

    a: set of integer
    t: set of integer
    d: set of integer
    h: set of integer

    Agenti: array(a) of integer
    Task: array(t) of integer
    Dimensioni: array(d) of string
    Proprieta: array (t,d) of integer
    Obiettivo: array (a,d) of integer
    w: array (d) of real
    Tipologia: array (t,h) of integer
    Capacita: array (a,h) of integer
    y: array(t,a) of mpvar
    s: array (a,d) of mpvar

end-declarations

initializations from "75-5dataset1.txt"   ! è il file che contiene i dati

    Agenti
    Task
    Dimensioni
    Proprieta

end-initializations

finalize(a)
finalize(t)
finalize(d)

forall(i in a, j in d) do
    Obiettivo(i,j) := integer(0.2 * sum(k in t) Proprieta(k,j))
end-do

forall(i in d) do
    w(i) := 1000 / sum(q in t) Proprieta(q,i)
end-do

FO:= sum(i in a, j in d) w(j) * s(i,j)

forall (i in t ) do          !vincolo (2.1)
    sum( j in a) y(i,j)=1
end-do

forall (i in a, j in d ) do           !vincolo (2.2)
        s(i,j)>= (sum(k in t)( y(k,i) * Proprieta(k,j)))-Obiettivo(i,j)
```

```
end-do

forall (i in a, j in d) do          !vincolo (2.3)
    s(i,j)>= Obiettivo(i,j)-(sum(k in t)( y(k,i) * Proprieta(k,j)))

end-do

forall (i in a, j in t ) do  !vincolo (2.4)
    y(j,i) <=1
end-do


forall(i in t, j in a) do
    yto0(i,j) := y(i,j) = 0
    yto1(i,j) := y(i,j) = 1
    sethidden(yto0(i,j),true)
    sethidden(yto1(i,j),true)
end-do

setparam("XPRS_MAXTIME", 3600)
exportprob(EP_MIN, "out.lp", FO)
start_time:=time(SYS_NOW)

flag:=1
while(flag=1)do
    flag:=0
    minimize(FO)
    tmax:=0
    amax:=0
    currentmax:= 0.0

    forall(i in t, j in a)do
        if(getsol(y(i,j))=0)then
            sethidden(yto0(i,j), false)
        end-if
        if(getsol(y(i,j))=1)then
            sethidden(yto1(i,j), false)
        end-if
    end-do

    forall(i in t, j in a) do
        if(getsol(y(i,j))>0 and getsol(y(i,j))<1)then
            flag:=1
            if(currentmax<getsol(y(i,j))) then
                currentmax:=getsol(y(i,j))
                tmax:=i
                amax:=j
            end-if
        end-if
    end-do
    if (flag=1)then
            sethidden(yto1(tmax,amax), false)
    end-if
end-do

! Integer problem

forall(i in t, j in a) do
    y(i,j) is_binary
end-do

minimize(FO)
current_of := getobjval;
```

```
currentmin:= getobjval
taskdim:=getsize(t)
writeln("La soluzione prima del while vale=", getobjval)

! Improvement con Multistart

writeln("La soluzione INIZIALE vale = ", getobjval)
forall(g in 1..10) do
    writeln("Ciclo =", g)
    noimprovements := 0

while(noimprovements < 3) do                    !criterio di STOP: mi fermo dopo
3 cicli che non portano a miglioramenti della soluzione
    f := integer(round((taskdim*random)+0.5))   !generazione random del
sottoinsieme di variabili
    if(f<taskdim-9)then
        fminimo:=f
        fmassimo:=fminimo+10
    end-if
    if(f>=taskdim-10)then
        fmassimo:=f
        fminimo:=fmassimo-10
    end-if

    forall(i in t, j in a) do
        if (getsol(y(i,j)) = 0) then
            sethidden(yto0(i,j), false)
            sethidden(yto1(i,j), true)
        end-if
        if (getsol(y(i,j)) = 1) then
            sethidden(yto0(i,j), true)
            sethidden(yto1(i,j), false)
        end-if
    end-do

    forall(i in t, j in a | i>fminimo and i<fmassimo ) do
        sethidden(yto0(i,j),true)
        sethidden(yto1(i,j),true)
    end-do

    minimize(FO)

    writeln(".....................................")
    writeln("La soluzione vale=", getobjval)

    if (getobjval < current_of) then
        writeln("La soluzione piu' recente e' minore quindi e' migliorata e
vale=", getobjval )
        current_of := getobjval
        noimprovements := 0
        writeln("quindi sovrascrivo questa soluzione", getobjval, "e il numero
di insuccessi viene azzerato infatti e' =", noimprovements)
        else
        noimprovements := noimprovements + 1;
    writeln("La soluzione piu' recente e' peggiorata, quindi rimane quella
precedente che vale=",current_of)
    writeln( "è aumentato il numero di non miglioramenti= ", noimprovements)
    end-if
end-do
    if(current_of < currentmin) then
    currentmin:=current_of
    writeln("Il currentmin vale = ", currentmin)
    end-if
```

```
end-do

end_time:= time(SYS_NOW)
writeln(" run time= ",end_time-start_time," ms ")
writeln(" la FO vale ", getobjval )

end-model
```

Appendice B

Excel data

Di seguito si riportano i principali dati Excel ottenuti dai lanci dei modelli in appendice A

CASO	Dataset	Xpress			Cplex		
		FO	Runtime [s]	Gap del solver	FO	Runtime [s]	Gap del solver
75-5	1	18,366	3287,22	0.00%	18,366	249,06	0.00%
	2	11,848	1342,05	0.00%	11,8478399	469,63	0.00%
	3	11,121	720,96	0.00%	11,121	1966,55	0.00%
	4	12,946	679,00	0.00%	12,946	169,36	0.00%
	5	5,629	2137,04	0.00%	5,629	1160,49	0.00%
100-5	1	13,238	1005,34	0.00%	13,238	281,13	0.00%
	2	1,138	566,30	0.00%	1,138	875,58	0.00%
	3	5,453	242,88	0.00%	5,453	978,03	0.00%
	4	17,908	843,08	0.00%	17,908	45,99	0.00%
	5	13,982	779,15	0.00%	13,982	119,75	0.00%
250-5	1	4,982	3600,02	1.22%	4,921	432,55	0.00%
	2	5,295	2522,83	0.00%	5,295	559,47	0.00%
	3	3,800	1391,08	0.00%	3,800	99,08	0.00%
	4	1,819	3599,58	6.77%	1,695	298,77	0.00%
	5	3,177	195,17	0.00%	3,177	290,08	0.00%
75-8	1	1.786,775	0,06	0.00%	1786,775	0,16	0.00%
	2	1.787,181	0,05	0.00%	1787,181	0,2	0.00%
	3	1.757,476	0,05	0.00%	1757,476	0,2	0.00%
	4	1.764,224	0,04	0.00%	1764,224	0,09	0.00%
	5	1.780,784	0,05	0.00%	1780,784	0,3	0.00%
100-8	1	1.792,022	0,05	0.00%	1792,022	0,17	0.00%
	2	1.792,880	0,07	0.00%	1792,880	0,19	0.00%
	3	1.785,795	0,06	0.00%	1785,795	0,09	0.00%
	4	1.778,603	0,06	0.00%	1778,603	0,2	0.00%
	5	1.792,548	0,07	0.00%	1792,548	0,09	0.00%
250-8	1	1.797,049	0,10	0.00%	1797,049	0,22	0.00%
	2	1.794,301	0,11	0.00%	1794,301	0,2	0.00%
	3	1.799,538	0,11	0.00%	1799,538	0,23	0.00%
	4	1.790,940	0,15	0.00%	1790,940	0,31	0.00%
	5	1.791,123	0,12	0.00%	1791,123	0,23	0.00%

Figura B.1: Risultati del modello approssimato

CASO	Dataset	NO-IMPROVEMENT < 3				NO-IMPROVEMENT < 2				NO-IMPROVEMENT < 10			
		FO	Media FO	Runtime [s]	Media Runtime	FO	Media FO	Runtime [s]	Media Runtime	FO	Media FO	Runtime [s]	Media Runtime
75-5	1	31,753	35,233	1,891	1,694	38,828	40,511	1,252	1,141	25,261	26,950	4	4,420
		38,656		1,861		30,975		1,25		27,916		5,015	
		38,537		1,451		51,623		1,033		25,493		3,172	
		31,985		1,571		40,616		1,028		29,128		5,492	
	5	25,472	26,539	1,546	1,500	20,259	21,222	1,26	1,409	20,098	15,406	2,602	3,732
		24,793		1,045		25,525		1,284		15,719		3,929	
		30,030		1,623		15,537		1,244		13,217		6,309	
		25,862		1,787		23,568		1,848		12,591		2,087	
100-5	1	20,493	20,978	1,754	1,484	20,629	25,424	0,912	1,002	24,444	21,879	2,575	2,822
		18,961		1,238		21,720		1,549		18,686		4,254	
		16,338		1,484		32,244		0,733		27,744		2,017	
		28,119		1,459		27,102		0,812		16,643		2,442	
	4	30,346	29,423	1,881	1,960	38,446	32,635	1,201	1,580	24,171	23,656	4,615	5,232
		32,756		1,548		33,962		1,258		25,450		4,894	
		25,773		2,275		29,170		2,311		22,274		6,272	
		28,817		2,136		28,963		1,548		22,730		5,148	
250-5	1	9,092	11,131	2,102	2,113	10,526	10,311	2,113	1,865	7,134	7,046	4,001	4,227
		10,782		2,413		8,800		1,728		8,241		3,483	
		15,558		1,119		9,700		2,058		7,268		4,095	
		9,092		2,816		12,217		1,562		5,541		5,329	
	4	9,772	7,551	2,304	2,431	9,027	10,553	1,354	2,010	4,714	4,607	6,015	5,503
		4,590		2,559		14,577		1,721		4,038		7,015	
		6,933		2,795		6,377		3,531		4,592		4,295	
		8,907		2,067		12,233		1,435		5,085		4,686	

Figura B.2: Risultati con algoritmo di miglioramento della soluzione e parametro no-improvement pari a 2, 3 e 10

CASO	Dataset	Multi-start 10		Multi-start 20		Multi-start 30	
		FO	Runtime [s]	FO	Runtime [s]	FO	Runtime [s]
75-5	1	27,109	3,419	25,291	8,109	21,223	7,186
	5	23,044	4,493	17,205	5,737	17,077	10,254
100-5	1	19,236	3,340	15,691	6,385	17,053	7,484
	4	23,024	3,880	22,260	7,313	26,612	11,306
250-5	1	8,241	6,160	6,842	10,308	7,548	13,424
	4	5,085	6,866	5,392	10,081	4,222	9,835

Figura B.3: Risultati con algoritmo multi-start e differenti reiterazioni

Appendice C

Esempio di istanza

Si riporta a titolo di esempio un file di istanza che viene letto dal programma. Si tratta di un'istanza completa da 75 task e 5 agenti. Il file che la contiene si chiamerebbe 75-5dataset1.txt.

Agenti : [

(1) 1 (2) 2 (3) 3 (4) 4 (5) 5

]

Task : [

(1) 1 (2) 2 (3) 3 (4) 4 (5) 5 (6) 6 (7) 7 (8) 8 (9) 9 (10) 10 (11) 11 (12) 12 (13) 13 (14) 14 (15) 15 (16) 16 (17) 17 (18) 18 (19) 19 (20) 20 (21) 21 (22) 22 (23) 23 (24) 24 (25) 25 (26) 26 (27) 27 (28) 28 (29) 29 (30) 30 (31) 31 (32) 32 (33) 33 (34) 34 (35) 35 (36) 36 (37) 37 (38) 38 (39) 39 (40) 40 (41) 41 (42) 42 (43) 43 (44) 44 (45) 45 (46) 46 (47) 47 (48) 48 (49) 49 (50) 50 (51) 51 (52) 52 (53) 53 (54) 54 (55) 55 (56) 56 (57) 57 (58) 58 (59) 59 (60) 60 (61) 61 (62) 62 (63) 63 (64) 64 (65) 65 (66) 66 (67) 67 (68) 68 (69) 69 (70) 70 (71) 71 (72) 72 (73) 73 (74) 74 (75) 75

]

Dimensioni : [

(1) "km" (2) "viaggi" (3) "n.soste"

]

Proprieta : [

(1 1) 79 (1 2) 19 (1 3) 2

(2 1) 229 (2 2) 31 (2 3) 4

(3 1) 188 (3 2) 19 (3 3) 3

(4 1) 57 (4 2) 21 (4 3) 2

(5 1) 78 (5 2) 32 (5 3) 1

(6 1) 217 (6 2) 25 (6 3) 2

(7 1) 37 (7 2) 40 (7 3) 1

(8 1) 48 (8 2) 39 (8 3) 2

(9 1) 88 (9 2) 33 (9 3) 2

(10 1) 112 (10 2) 11 (10 3) 2

(11 1) 153 (11 2) 23 (11 3) 4

(12 1) 42 (12 2) 32 (12 3) 1

(13 1) 224 (13 2) 21 (13 3) 1

(14 1) 134 (14 2) 29 (14 3) 1

(15 1) 146 (15 2) 33 (15 3) 1

(16 1) 171 (16 2) 26 (16 3) 2

(17 1) 13 (17 2) 24 (17 3) 2

(18 1) 134 (18 2) 21 (18 3) 1

(19 1) 196 (19 2) 11 (19 3) 3

(20 1) 45 (20 2) 8 (20 3) 3

(21 1) 243 (21 2) 2 (21 3) 3

(22 1) 36 (22 2) 38 (22 3) 1

(23 1) 132 (23 2) 11 (23 3) 1

(24 1) 102 (24 2) 12 (24 3) 2

(25 1) 132 (25 2) 31 (25 3) 4

(26 1) 231 (26 2) 6 (26 3) 1

(27 1) 250 (27 2) 33 (27 3) 4

(28 1) 15 (28 2) 33 (28 3) 3

(29 1) 79 (29 2) 7 (29 3) 1

(30 1) 111 (30 2) 12 (30 3) 2

(31 1) 248 (31 2) 3 (31 3) 2

(32 1) 207 (32 2) 13 (32 3) 3

(33 1) 66 (33 2) 21 (33 3) 3

(34 1) 249 (34 2) 15 (34 3) 1

(35 1) 37 (35 2) 34 (35 3) 2

(36 1) 194 (36 2) 33 (36 3) 1

(37 1) 171 (37 2) 5 (37 3) 4

(38 1) 65 (38 2) 10 (38 3) 4

(39 1) 54 (39 2) 25 (39 3) 2

(40 1) 125 (40 2) 7 (40 3) 4

(41 1) 166 (41 2) 38 (41 3) 3

(42 1) 251 (42 2) 18 (42 3) 3

(43 1) 67 (43 2) 34 (43 3) 1

(44 1) 211 (44 2) 9 (44 3) 1

(45 1) 59 (45 2) 12 (45 3) 2

(46 1) 75 (46 2) 11 (46 3) 3

(47 1) 185 (47 2) 1 (47 3) 4

(48 1) 205 (48 2) 3 (48 3) 2

(49 1) 157 (49 2) 39 (49 3) 3

(50 1) 243 (50 2) 14 (50 3) 3

(51 1) 206 (51 2) 15 (51 3) 4

(52 1) 89 (52 2) 30 (52 3) 2

(53 1) 147 (53 2) 4 (53 3) 3

(54 1) 55 (54 2) 3 (54 3) 3

(55 1) 204 (55 2) 7 (55 3) 1

(56 1) 101 (56 2) 38 (56 3) 4

(57 1) 202 (57 2) 27 (57 3) 4

(58 1) 53 (58 2) 22 (58 3) 3

(59 1) 193 (59 2) 39 (59 3) 2

(60 1) 19 (60 2) 23 (60 3) 2

(61 1) 123 (61 2) 13 (61 3) 1

(62 1) 109 (62 2) 9 (62 3) 2

(63 1) 71 (63 2) 36 (63 3) 1

(64 1) 195 (64 2) 37 (64 3) 3

(65 1) 137 (65 2) 30 (65 3) 2

(66 1) 62 (66 2) 14 (66 3) 4

(67 1) 51 (67 2) 38 (67 3) 4

(68 1) 206 (68 2) 23 (68 3) 2

(69 1) 148 (69 2) 36 (69 3) 3

(70 1) 123 (70 2) 35 (70 3) 1

(71 1) 177 (71 2) 24 (71 3) 3

(72 1) 148 (72 2) 32 (72 3) 1

(73 1) 121 (73 2) 22 (73 3) 1

(74 1) 37 (74 2) 21 (74 3) 1

(75 1) 170 (75 2) 5 (75 3) 3

]

Capacita : [

(1 1) 54 (1 2) 6 (1 3) 40 (1 4) 95 (1 5) 66 (1 6) 47

(2 1) 52 (2 2) 28 (2 3) 4 (2 4) 100 (2 5) 92 (2 6) 32

(3 1) 71 (3 2) 91 (3 3) 80 (3 4) 74 (3 5) 42 (3 6) 39

(4 1) 67 (4 2) 75 (4 3) 72 (4 4) 89 (4 5) 95 (4 6) 24

(5 1) 14 (5 2) 34 (5 3) 80 (5 4) 54 (5 5) 7 (5 6) 39

]

Bibliografia

[1] Billing C., Jaehn F., Wensing T., Annals Operations Research (2018), *Fair task allocation problem*
https://doi.org/10.1007/s10479-018-3052-3

[2] F. Facchinei, C. Mannino, S. Lucidi, M. Roma, *Appunti delle lezioni di Ricerca Operativa*, Università di Roma, Sede di Latina (università Pontina), Corso di Laurea in Ingegneria Informatica, a.a. 2003-2004.

[3] G. Bigi, A. Frangioni, G. Gallo, S. Pallottino, M. G. Scutellà, *Appunti di Ricerca Operativa*, SEU - Servizio Editoriale Universitario di Pisa, Corso di Laurea in Ingegneria Informatica, a.a. 2018-2019

[4] Wikipedia L'enciclopedia libera *"Classi di complessità P e NP"*
https://it.wikipedia.org/wiki/Classi_di_complessita'_P_e_NP

[5] A. Agnetis, C. Meloni, *Appunti sui metodi metaeuristici di ricerca*

[6] Marco A. Boschetti, Vittorio Maniezzo, *"Combining Exact Methods and Heuristics"*, Wiley Enciclopedia of Operations Research and Management Science, 2011

[7] *Xpress-Mosel Language Reference Manual*
https://www.msi-jp.com/xpress/learning/square/application.pdf

Ringraziamenti

Arrivati alla conclusione di questo lungo percorso ritengo doveroso dedicare uno spazio del mio elaborato alle persone che hanno contribuito, con il loro supporto, alla realizzazione dello stesso e al raggiungimento di questo traguardo.
In primis, un ringraziamento al mio relatore Professore Marco Ghirardi, per la sua pazienza, per i suoi consigli e per le conoscenze trasmesse durante il percorso di stesura dell'elaborato.
Ringrazio infinitamente i miei genitori e mia sorella che mi hanno sempre sostenuto, appoggiando ogni mia decisione.
Infine ringrazio tutti i miei amici e colleghi che hanno avuto un ruolo determinante nel conseguimento di questo risultato. Grazie per aver condiviso con me in questi anni un percorso così bello e importante.