Università degli Studi di Padova

REPORT DELL'ESERCIZIO

Travelling Salesman Problem (TSP)

Problema del commesso viaggiatore

Corso: Metodi e Modelli per l'ottimizzazione combinatoria

Docenti: Luigi De Giovanni, Marco Di Summa

Studente: Davide Meneghetti matricola 1133361

INDICE

1. Prologo

Descrizione del problema

Il problema da risolvere è il seguente:

> *A company produces boards with holes used to build electric frames. Boards are positioned over a machines and a drill moves over the board, stops at the desired positions and makes the holes. Once a board is drilled, a new board is positioned and the process is iterated many times. Given the position of the holes on the board, the company asks us to determine the hole sequence that minimizes the total drilling time, taking into account that the time needed for making an hole is the same and constant for all the holes.*

Dal punto di vista della programmazione lineare è utile vederlo come un caso di TSP (Travelling Salesman Problem) o Problema del Commesso Viaggiatore.

Strategie per lo studio del problema

Allo scopo di trovare una soluzione ammissibile al problema ho usato due strategie:

- Il metodo del simplesso, sfruttando in particolare la libreria CPLEX fornita da IBM.
- L'applicazione e la conseguente analisi dei risultati di funzioni euristiche, in particolare ho scelto di testare *Deterministic Anneling* e *Ant System*.

2. Generazione delle istanze

In modo che i test sulle euristiche fossero significativi ho ritenuto necessario generare diverse istanze in maniera pseudocasuale al variare della numerosità n di punti, mantenendo una dimensione 20x20 dell'area di lavoro. Le istanze consistono in un insieme di n coordinate intere che identificano la posizione sulla superficie a disposizione.

Approcci

Ho generato gli esempi seguendo due approcci:

- Random: i fori non seguono nessun ordine o schema particolare e le coordinate vengono generate pseudo-casualmente secondo una distribuzione uniforme;
- Settori: suddivide la superficie in 25 settori quadrati e ne sceglie 7 casualmente dove i punti verranno generati. In ognuno dei settori genero n/7 fori con approccio random. Ed infine genero casualmente gli eventuali punti mancanti senza badare alla suddivisione in settori. Questo approccio porta ad avere alcune zone dense di punti e altre vuote.

Distanze

Dopo aver generato le coordinare dei punti calcolo le distanze fra essi attraverso la distanza Manhattan che simula uno strumento che si sposta seguendo gli assi orizzontale e verticale; e attraverso la distanza Euclidea imitando una macchina con maggiore "capacità di movimento".

Quantità

Le istanze sono generate su dimensioni della superficie di lavoro pari a 20x20 considerando per semplicità solo punti con coordinate intere. Il numero di fori varia da un minimo di 10 ad un massimo di 100 con passo 10.

Per ogni esempio generato con i due approcci viene calcolata la distanza secondo i due criteri.

Esecuzione

Nella cartella *genaratoreIstanze* si trova il codice per creare le istanze nel modo appena descritto. Lanciando il comando *./generatoreIstanze* vengono creati automaticamente 40 istanze nell'omonima cartella.

3. Prima parte: CPLEX

Modellazione del problema

Il problema di perforazione può essere generalizzato rappresentandolo tramite un grafo G=(V,A): i vertici V rappresentano i punti in cui la macchina dovrà operare i fori, mentre gli archi (i, j) appartenenti all'insieme A rappresentano il tratto da percorrere dalla posizione i alla posizione j, e ad ognuno di essi verrà assegnato un peso c_{ij} che indica il tempo necessario alla punta per spostarsi, che corrisponde alla distanza fra i punti supponendo una velocità di movimento costante. Il problema richiede quindi di trovare il cammino hamiltoniano di costo minimo, cioè il percorso più breve che visita tutti i nodi.

Uso di CPLEX

Nella prima parte dell'esercizio era richiesto di implementare un algoritmo esatto utilizzando la libreria di CPLEX per modellare quindi risolvere il problema. A tale scopo ho dichiarato variabili e vincoli del problema attraverso la funzione *setupLP*, sfruttando la funzione *CHECKED_CPX_CALL* di libreria per "passarli" al risolutore di CPLEX.

Variabili e Strutture dati

Allo scopo di rendere più semplice la dichiarazione dei vincoli, visto che è determinante ricordare l'ordine di definizione delle variabili, l'ho memorizzato servendomi di una matrice *map* a tre dimensioni (N x N x 2). Nell'elemento *map[i][j][0]* viene salvato x_{ij} mentre in *map[i][j][1]* viene salvato y_{ij}.

Un piccolo accorgimento durante la definizione delle variabili, sia le x_{ij} sia le $y_{ij,}$ si ha quando non vengono "trasmesse" a CPLEX nell'istante in cui i=j perché superflue, in quanto sono le variabili

che si riferiscono ai "cappi" del grafo (cioè archi entranti ed uscenti da uno stesso nodo), variabili inutili perché non verrebbero sicuramente utilizzate dal solutore. Questo trick permette di operare con meno variabili: in particolare consente una diminuzione di 2N variabili del problema. Porta in dote anche un leggero contro: la definizione dei vincoli viene "appesantita" con l'aggiunta di un indice e l'accorgimento dei valori degli altri, rendendo la leggibilità e l'interpretazione meno immediata.

Esecuzione

Il codice relativo all'esercizio risolto con CPLEX si trova nella cartella *cplexSol*. Per testare il solutore di CPLEX e far risolvere una determinata istanza del problema, basta specificarne il nome scrivendo ad esempio: *./main ../istanze/CE30.dat* .

4. Seconda parte: euristiche

Descrizione delle euristiche

Le funzioni euristiche sono dei metodi, non necessariamente esatti, che non garantiscono quindi di raggiungere sempre il risultato ottimo in assoluto, ma il loro obiettivo è di raggiungere mediamente un risultato buono. Esistono euristiche costruttive che costituiscono una soluzione partendo da zero e altre dette migliorative che operano una ricerca partendo da una soluzione ammissibile e cercando appunto di migliorarla.

Nel problema di TSP preso in esame ho scelto di applicare e confrontare due funzioni euristiche che si basano su concetti differenti: Deterministic Anneling e Ant System. Entrambe anno qualche aspetto pseudo-casuale, quindi in fase di test le istanze vengono provate 10 volte ognuna e vengono memorizzati i valori medi di tempi e valori ottimi raggiunti.

Idee teoriche

Deterministic Anneling

Deterministic Anneling (da ora in poi indicherò con "DA") è un'euristica migliorativa che opera una ricerca locale (entro un numero massimo di iterazioni impostabile) allo scopo di trovare uno scambio di archi del cammino hamiltoniano che migliori il valore della funzione obiettivo. Inoltre si può scegliere fra due possibilità di accettazione del nuovo ottimo: nella prima ad ogni iterazione t, la soluzione x_{t+1} viene accettata se $f(x_{t+1}) < f(x_t) + \theta 1$, dove $\theta 1$ è un parametro controllato dall'utente; nella seconda all'iterazione t, la soluzione x_{t+1} è accettata se $f(x_{t+1}) < \theta 2\ f(x_t)$, dove $\theta 2$ è un parametro regolabile dall'utente, in genere impostato su un valore di poco superiore ad 1. Quest'ultima opzione è quella che ho scelto di applicare.

Ant System

Ant System (o Ant Colony Optimisation, da ora in poi indicherò con "AS") è nata e deve il nome dall'osservazione del comportamento delle formiche, che in particolare durante la ricerca del cibo secernono lungo il cammino una sostanza riconoscibile a tutte le appartenenti della colonia: il feromone. La presenza di questa sostanza costituisce un'informazione per le altre formiche, che vi sono attratte; la quantità di feromone presente su un percorso dipende dalla sua lunghezza e dalla qualità della fonte di cibo raggiungibile percorrendolo. Con il passare del tempo i cammini più interessanti, cioè quelli che portano a fonti di cibo migliori, diventano sempre più frequentati e in essi è depositata una maggiore quantità di feromone.

Nel caso in oggetto ad ogni arco sono assegnate una visibilità, che consiste in un valore costante che corrisponde all'inverso della lunghezza dello stesso, e una traccia di feromone, il cui valore è aggiornato dinamicamente in base al procedere dell'algoritmo; la somma di queste due grandezze costituisce una misura di desiderabilità dell'arco.

Nella pratica, ad ogni iterazione l'ideale formica parte da un vertice casuale e costruisce un cammino hamiltoniano con un approccio greedy: ad ogni nodo sceglie l'arco con desiderabilità maggiore. Al termine di ogni iterazione le tracce di feromone vengono aggiornate: vengono ridotti i valori delle tracce di una frazione compresa fra 0 e 1, e lungo il cammino appena trovato vengono ridistribuite nuove quantità pari all'inverso della sua lunghezza. La dispersione del feromone serve ad evitare percorsi cattivi, come tipicamente quelli delle prime iterazioni, che potrebbero condizionare negativamente la ricerca successiva.

Deterministic Anneling

Inizializzazione

L'euristica è di tipo migliorativo quindi esegue una ricerca locale a partire da una soluzione ammissibile che per evitare che le soluzioni siano molto simili è generata casualmente.

Ricerca e criterio di accettazione

All'i-esima iterazione l'euristica valuta n-i coppie di archi casuali individuando quella che migliora maggiormente la funzione obiettivo e conseguentemente opera lo scambio. Se il valore della soluzione corrente migliora quello "modificato"(meno restrittivo dell'originale, perché moltiplicato per teta) della soluzione di riferimento allora la soluzione appena trovata diventa il nuovo riferimento.

Terminazione

L'euristica si ferma una volta raggiunto il numero massimo di iterazioni e restituisce la miglior soluzione individuata nel corso di tutte le iterazioni.

Esecuzione

Il codice relativo all'implementazione di Deterministic Anneling si trova nella cartella *DA.* Per risolvere una determinata istanza del problema utilizzando tale euristica sono richieste un numero massimo di iterazioni e un indice θ vicino e maggiore a 1, vanno quindi specificati come nell'esempio: *./main ../istanze/ CE30.dat 30 1.10* .

Ant System

Inizializzazione e costruzione della soluzione

Ad ogni iterazione la "formica" costruisce un percorso scegliendo man mano l'arco più desiderabile. E' immediato capire che se il vertice di partenza fosse sempre lo stesso le soluzioni risulterebbero

tutte uguali, quindi ogni volta la costruzione del cammino parte da un vertice casuale.

Bonus nuovo cammino migliore

Ho scelto di cospargere gli archi di una quantità aggiuntiva di feromone quando viene trovata una soluzione migliore della corrente, per fare in modo che le informazioni utili "guadagnate" all'ultima iterazione non venissero tralasciate.

Terminazione

L'euristica si ferma una volta raggiunto il numero massimo di iterazioni e restituisce la miglior soluzione individuata.

Esecuzione

Il codice relativo all'implementazione di Ant System si trova nella cartella *Ant.* Per risolvere una determinata istanza del problema utilizzando tale euristica sono richieste un numero massimo di iterazioni e un indice ro compreso fra 0 e 1, vanno quindi specificati come nell'esempio: *./main ../istanze/CE30.dat 30 0.8* .

Scelta dei Parametri

Dopo aver osservato il comportamento delle euristiche con istanze di dimensioni diverse ho scelto di settare il numero massimo di iterazioni per entrambe le euristiche pari al numero di fori da operare. Questa determinata soglia permette ad AS di convergere ad una buona soluzione e nel caso di DA di operare un buon numero di tentativi. Inoltre per quanto riguarda DA ho settato $\theta=1.10$ mentre in AS il coefficiente di dispersione di feromone ro=0.8. Queste scelte sono tali per permettere a entrambe le euristiche, ognuna in base al criterio che usa, di trovare soluzioni migliorative senza seguire strade e soluzioni "poco buone".

5. Risultati dei test e analisi

Analisi valore ottimo e analisi tempo

Dai test effettuati ho raccolto i dati relativi al valore ottimo ottenuto con CPLEX, i tempi e i valori ottimi raggiunti mediamente dalle euristiche.

Ho successivamente comparato i dati ottenuti dalle euristiche rispetto a quelli ottenuti facendo risolvere le istanze a CPLEX.

Visto che alcune istanze impiegavano molto tempo ho impostato un limite di tempo per trovare una soluzione pari a 240 secondi, questa scelta porta, tipicamente le istanze più grandi, ad ottenere non ottenere soluzioni o raggiungerne di sub-ottime. Per questo motivo nelle tabelle che propongo una parte dei risultati risultano anomali: in particolare gli scarti dei valori raggiunti dalle euristiche dal valore ottimo ottenuto con CPLEX e il paragone fra relativi tempi di elaborazione.

Dati valore ottimo

In tabella 1 riporto gli scostamenti medi dal valore ottenuto con CPLEX in percentuale. Distinguo valori ottenuti da istanze create con approccio Random, a Settori, e "globale".

Faccio notare che i valori riferiti agli scostamenti con numerosità superiore a 50 sono viziati dal fatto che l'esecuzione di CPLEX raggiunge il limite di tempo, di conseguenza non raggiunge l'ottimo assoluto. Nel caso in cui CPLEX non abbia ottenuto nemmeno una soluzione entro il limite di tempo ho scelto di impostare il valore ottimo pari al migliore prodotto delle euristiche. Questa anomalia spiega perché alcuni valori raggiunti dall'euristica AS risultino migliori di quelli ottenuti da cplex.

	Random		Settori		Globale	
n	DA	AS	DA	AS	DA	AS
10	39,15%	0,00%	31,49%	3,12%	35,32%	1,56%
20	28,01%	17,58%	56,28%	0,33%	42,14%	8,95%
30	51,96%	7,63%	56,13%	3,65%	54,05%	5,64%
40	59,13%	5,46%	63,26%	10,16%	61,19%	7,81%
50	66,03%	13,96%	71,69%	14,24%	68,86%	14,10%
60	64,38%	7,29%	63,20%	-0,91%	63,79%	3,19%
70	59,72%	-16,12%	67,36%	-2,47%	63,54%	-9,30%
80	58,55%	-33,54%	69,53%	-0,74%	64,04%	-17,14%
90	64,96%	-0,02%	65,31%	-18,35%	65,13%	-9,19%
100	61,08%	-14,15%	69,89%	0,00%	65,48%	-7,07%

Tabella 1: Scostamenti medi dal valore ottimo individuato da CPLEX in percentuale

Nel grafico 1 che segue espongo gli scostamenti percentuali con numerosità che varia da 10 a 50, escludendo valori viziati dal limite di tempo. Si nota che l'euristica AS ottiene risultati migliori di DA, molto più vicini a quelli calcolati da CPLEX.

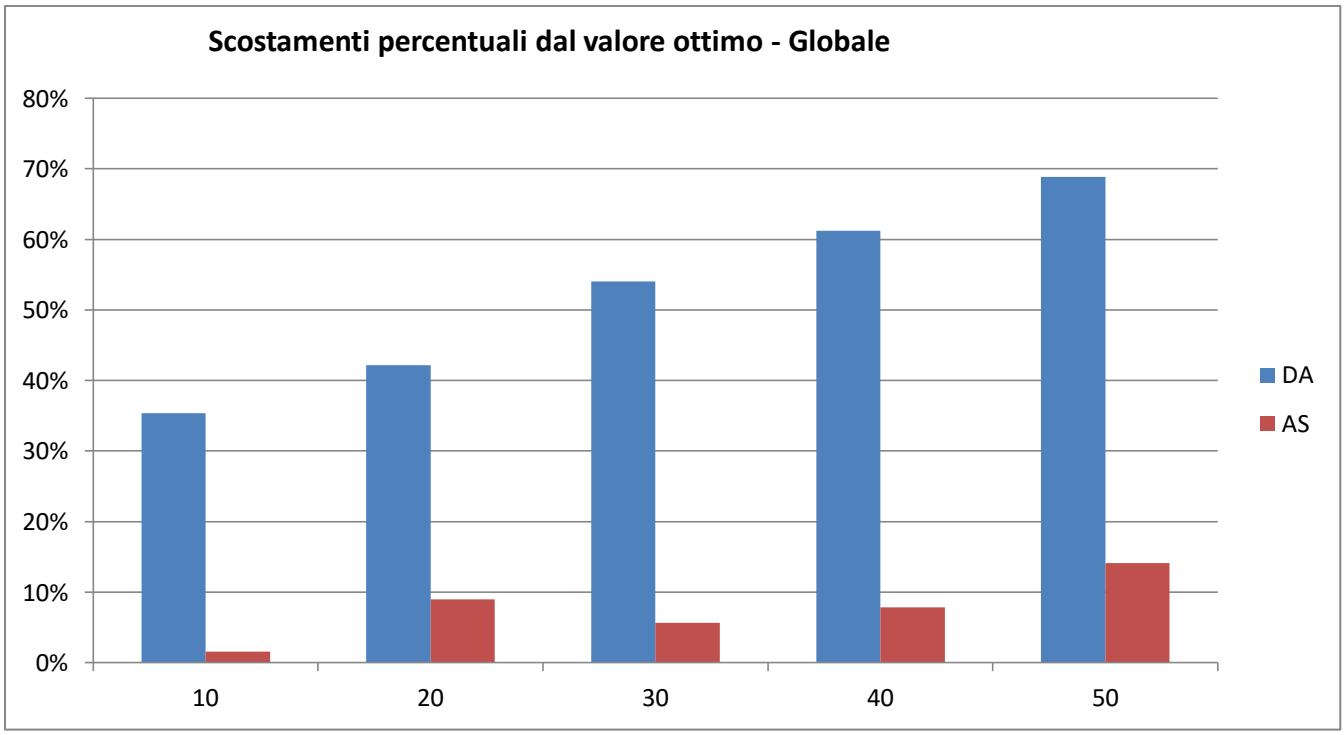

Grafico 2: Istogramma degli scostamenti medi dal valore ottenuto da CPLEX in percentuale al variare della numerosità.

Dati tempo

Nella seguente tabella 2 espongo i tempi medi di elaborazione dei vari metodi di soluzione delle istanze suddividendo le categorie in maniera analoga alla tabella precedente.

	Random			Settori			Globale		
n	CPLEX	DA	AS	CPLEX	DA	AS	CPLEX	DA	AS
10	0,148	0,000	0,001	0,083	0,000	0,001	0,115	0,000	0,001
20	0,933	0,001	0,001	1,873	0,001	0,001	1,403	0,001	0,001
30	9,550	0,001	0,001	17,793	0,001	0,002	13,671	0,001	0,002
40	44,681	0,001	0,002	31,315	0,001	0,004	37,998	0,001	0,003
50	71,965	0,002	0,002	128,679	0,002	0,005	100,322	0,002	0,004
60	212,205	0,002	0,003	240,000	0,002	0,007	226,102	0,002	0,005
70	240,000	0,003	0,004	240,000	0,003	0,009	240,000	0,003	0,006
80	240,000	0,003	0,004	240,000	0,003	0,012	240,000	0,003	0,008
90	240,000	0,004	0,005	240,000	0,004	0,016	240,000	0,004	0,011
100	240,000	0,004	0,006	240,000	0,004	0,021	240,000	0,004	0,014

Tabella 2: tempi medi di elaborazione in secondi

Si nota, come già indicato, che le istanze con numerosità pari e superiore a 60 raggiungono il limite di tempo massimo di 240 secondi. Dal Grafico 2 che segue vediamo il variare del tempo di elaborazione necessario a CPLEX per risolvere le istanze con numerosità da 10 a 60.

Inoltre si osserva come all'aumentare della numerosità il tempo di elaborazione per CPLEX aumenti significativamente rispetto al tempo necessario alle euristiche. Quest'ultime infatti mantengano un tempo molto più esiguo al variare della dimensione dell'istanza come mostro nel grafico 3.

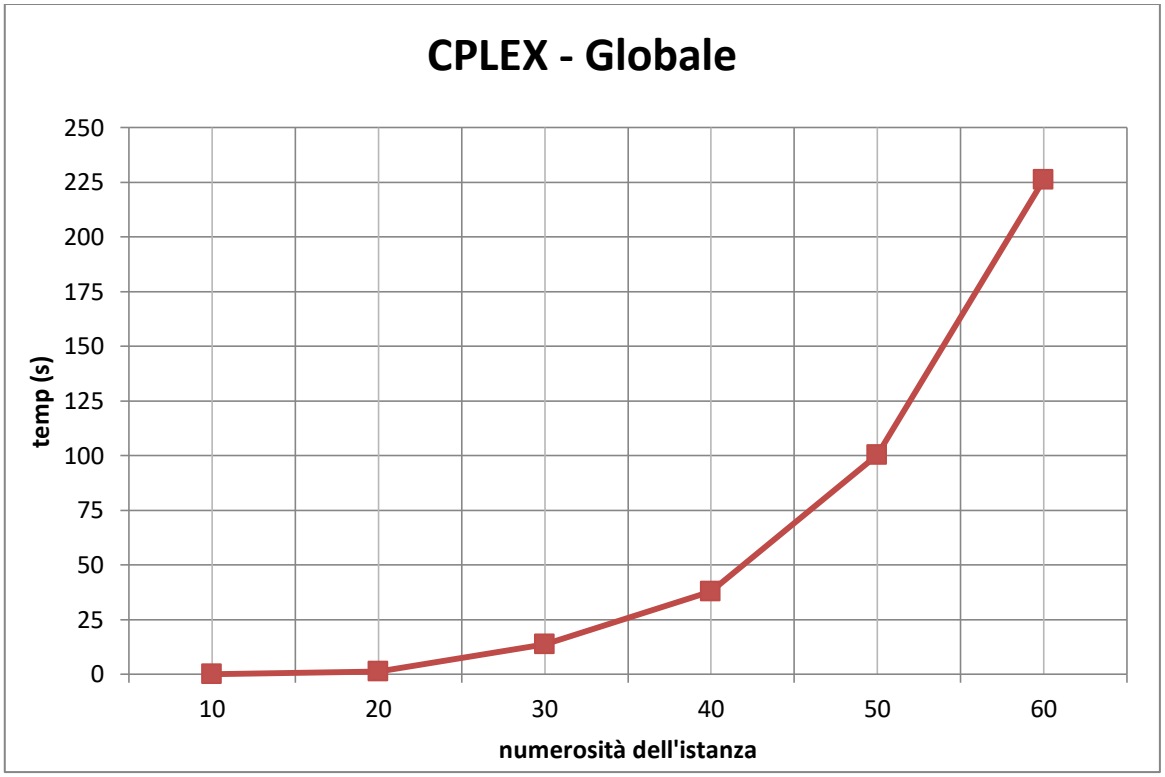

Grafico 2: tempo medio (in secondi) di elaborazione da parte di CPLEX al variare della dimensione dell'istanza.

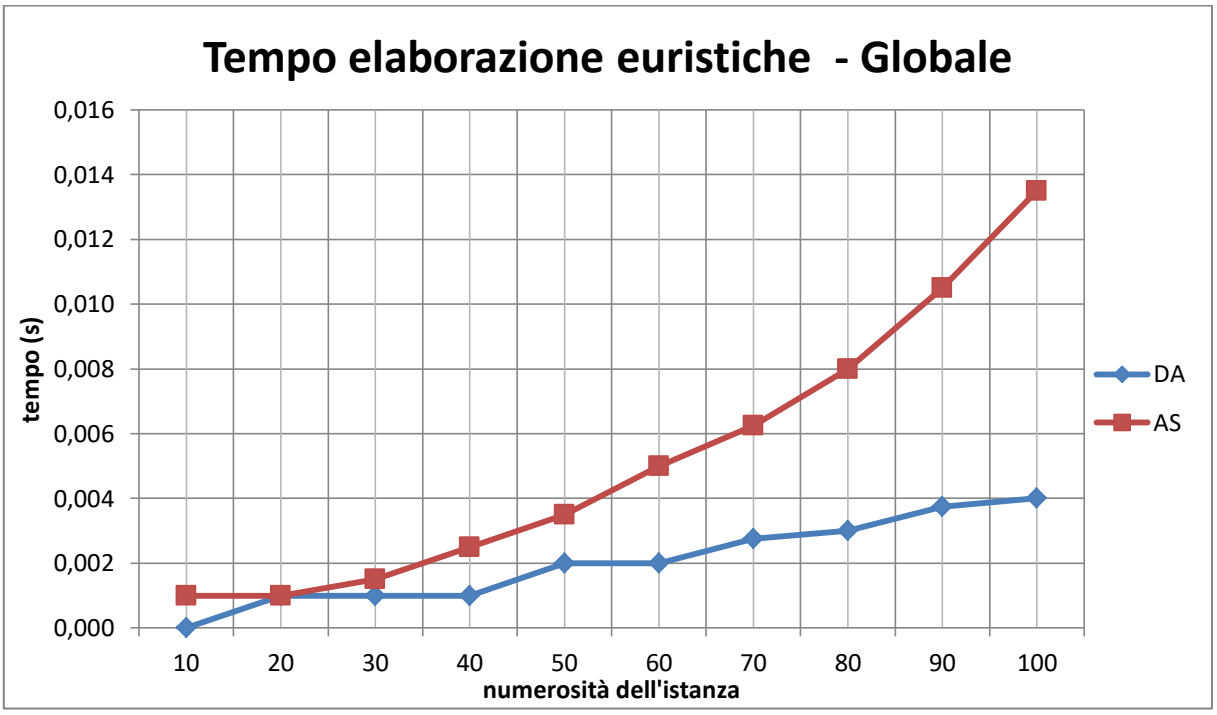

Grafico 3: tempo medio (in secondi) di elaborazione da parte delle euristiche al variare della dimensione dell'istanza.

6. Conclusioni

Dai risultati ottenuti dalla prove si evince che:

- Per istanze di valore limitato (n <= 50) CPLEX ottiene un valore ottimo esatto in un tempo ragionevole;
- Per istanze di dimensione discreta CPLEX richiede un tempo cospicuo per risolvere il problema e impostando un limite di tempo troppo basso si rischia di non ottenere nemmeno una soluzione;
- L'euristica DA opera l'elaborazione in un tempo molto ridotto, ma i risultati ottenuti sono deludenti perché mediamente sono ben lontani dal valore ottimo individuato da CPLEX;
- L'euristica AS elabora il problema in poco tempo, nonostante ne impieghi una quantità leggermente superiore a DA. I valori raggiunti da tale euristica sono buoni (con istanze di 50 punti lo scarto medio è minore del 15%).

In generale CPLEX rimane la scelta migliore per raggiungere il valore ottimo globale con istanze di dimensioni non troppo grandi. Quando invece il solutore sviluppato da IBM deve fare i conti con i limiti di tempo e la richiesta è tale che è possibile accettare una soluzione sufficientemente buona in un periodo più ragionevole, allora l'euristica Ant System sembra essere una valida alternativa.