Esercitazione di laboratorio - Parte II

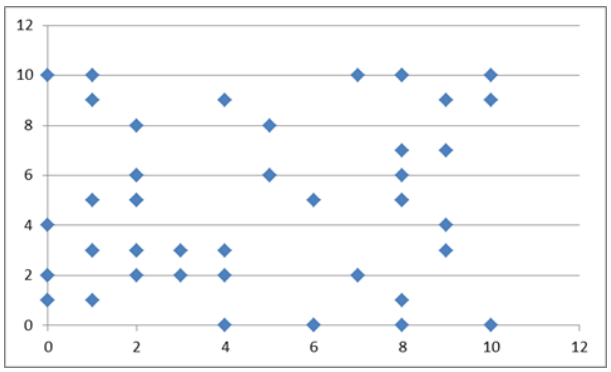
Introduzione

Nella seconda parte dell'esercitazione si vuole costruire un metodo alternativo per il problema presentato nella parte precedente tramite l'uso delle metaeuristiche. Brevemente, il problema consisteva nel trovare il percorso minimo per permettere di perforare una lastra nel più breve tempo possibile. Il problema è dunque riconducibile al trovare il più breve ciclo hamiltoniano di un grafo completo.

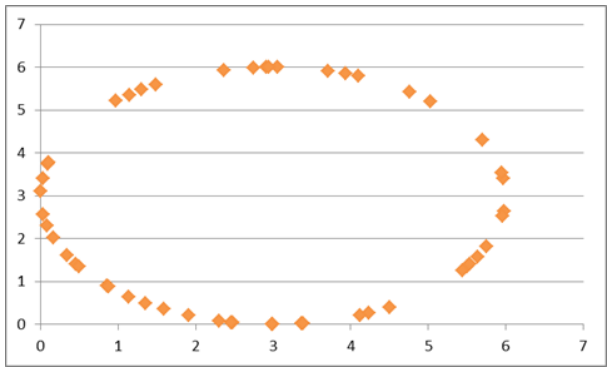
Disposizione dei nodi

L'algoritmo è stato utilizzato con una quantità variabile di punti. Questa quantità varia molto ed è visibile nella variabile *static const int* nel file *TSP.h*. Questo per poter vedere il comportamento del programma sia in presenza di pochi che di tanti nodi. Ovviamente, i nodi sono stati rappresentati come nella precedente parte dell'esercitazione (coordinate x e y).
Anche le disposizioni di tali punti sono rimaste invariate. Questo per poter confrontare tempi e risultati ottenuti con l'esercitazione precedente. Le disposizioni scelte per l'esercitazione sono:

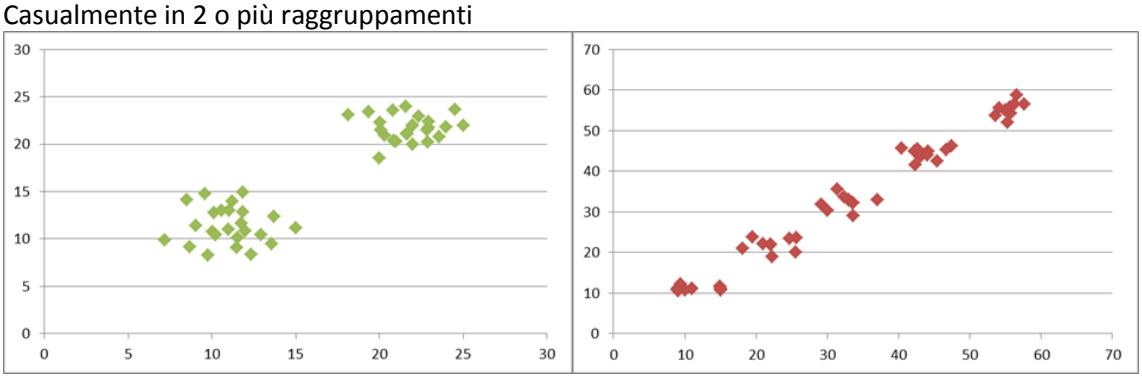
- Completamente casuale



- Casualmente lungo una circonferenza



- Casualmente in 2 o più raggruppamenti



Per le ultime due sono state utilizzate le coordinate polari, quindi la generazione è molto veloce.

Implementazione

Per una corretta esecuzione del programma è necessario conoscere alcuni parametri (tutti obbligatori, ma non necessariamente utilizzati):

- Numero dei nodi da utilizzare

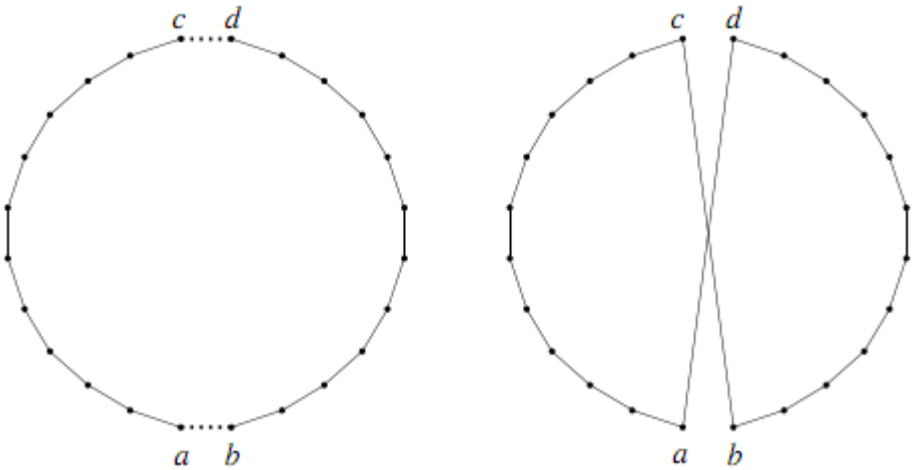
- Disposizione dei punti (0=4 punti, 1=random, 2=circonferenza, 3=clusters, 4=leggi da file)
- Euristica (0=Local Search, 1=Tabu Search, 2=Tabu Search con Aspirazione)
- Tabu Length (serve solo se il terzo parametro equivale a 1 o 2)
- Massimo numero di iterazioni (serve solo se il terzo parametro equivale a 1 o 2)
- Numero di cluster (serve solo se il secondo parametro equivale a 3)
- Tipo di soluzione iniziale (0=random, 1=Nearest Neighbour)
- K-opt (0=2-opt, 1=3-opt)
- Diversificazione (0=con diversificazione, 1=senza diversificazione)

Ricerca della soluzione

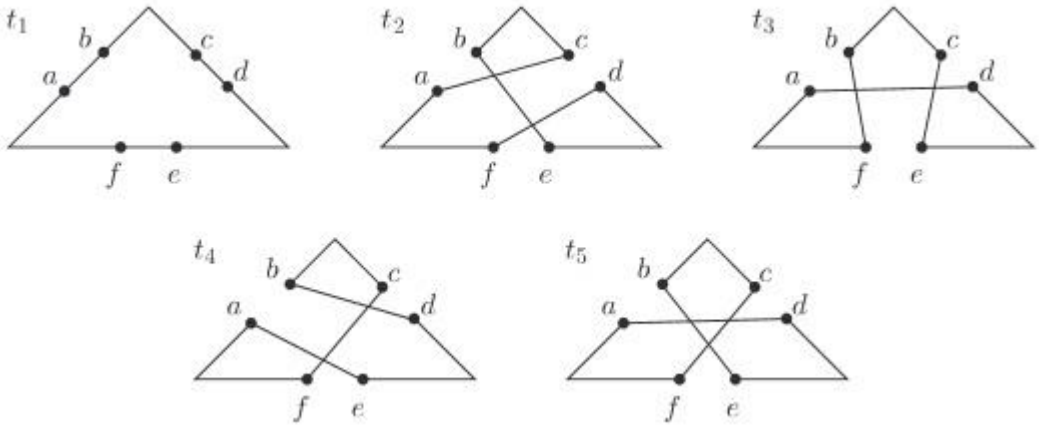
Per la ricerca del vicinato è stato utilizzato il seguente schema:

- Soluzione iniziale: la soluzione di partenza è stata generata casualmente. Questo perché non sempre posso raggiungere l'ottimo scegliendo sempre il vicino migliore. In questo modo, posso utilizzare più punti di partenza e avere una maggiore probabilità di avvicinarmi all'ottimo.
  Come alternativa è stata costruita una soluzione iniziale scegliendo il migliore dei nodi rimanenti (Nearest Neighbour).
- Rappresentazione soluzione: data la semplicità, una soluzione è stata rappresentata dalla sequenza del ciclo (0-1-5-6-7-0). I vantaggi offerti sono:
  - Scambi eseguiti in tempo O(1).
  - Ammissibilità di ogni sequenza (ripetendo solamente il nodo iniziale/finale) visto che il grafo è completo.
- Applicazione: la funzione scelta per trovare i membri del vicinato consiste nello scambio di 2 archi del ciclo con 2 esterni (2-opt) . Anche se eseguiti anch'esse in tempo costante, l'aggiunta e la rimozione di archi non sono una scelta sensata per il nostro problema. Si è scelta inoltre una funzione che generasse un numero limitato di vicini, in modo da rendere la loro esplorazione molto veloce. È stato realizzato anche l'algoritmo per il 3-opt per poter confrontare tempi e risultati. Per la realizzazione del 3-opt, si è preso un grafo generico (come quello nella figura sottostante) e si è visto che sono possibili solamente 4 scambi (al contrario dell'unico scambio che avviene con il 2-opt) . Tutte le altre combinazioni sono state scartate per i seguenti motivi:
  - Collegando due lettere "vicine e ma non direttamente connesse" (ad esempio *c* con *b*) si ottenevano dei sottocicli
  - Collegando due lettere "vicine e già connesse" (ad esempio *c* con *d*) si ottenevano dei semplici 2-opt e perciò non sono state considerate queste combinazioni

  È stato dunque trovato il pattern per i nuovi scorrimenti delle sequenze di nodi e si è di conseguenza modificato il metodo *swap* per poter ricostruire il ciclo correttamente in base a quale combinazione è stata scelta.



2-opt



3-opt con i 4 possibili scambi

- Esplorazione del vicinato: i metodi utilizzati per l'esplorazione del vicinato sono stati i seguenti:
  - Ricerca locale: analizza la funzione obiettivo di ogni vicino e mi sposto su un eventuale vicino migliore. In caso di assenza di vicini migliori, l'algoritmo termina.
  - Tabu search: l'algoritmo si ricorda le ultime mosse eseguite. È necessaria in quanto, non terminando più l'algoritmo in assenza di vicini migliori, passo al migliore dei peggioranti. Nell'iterazione successiva però tornerei alla soluzione precedente in quanto sarebbe la migliore, e così via. Ciò si può evitare ricordando le ultime mosse fatte, in modo da non ripeterle. È stata usata una tabu-length pari a 7, in quanto una tabu-length troppo corta non impedisce comunque i cicli, mentre una tabu-length troppo lunga potrebbe bloccare troppi vicini. Da notare che vengono salvate le mosse e non le soluzioni in quanto vorrei poter, se necessario, tornare ad una soluzione precedente per utilizzare un'altra strada.
  - Aspirazione: memorizzando mosse e non soluzioni, potrebbe succedere che la soluzione generata da una mossa tabu abbia delle caratteristiche che la rendano comunque interessante. Sarebbe quindi meglio considerare queste soluzioni nonostante siano state proibite.
- Valutazione: la valutazione dei vicini viene effettuata tramite confronto della funzione obiettivo (con il metodo scelto non posso avere soluzioni non ammissibili, quindi non è necessaria nessuna "penalizzazione"). In pratica viene c'è la funzione incaricata di controllare l'eventuale miglioramento della funzione obiettivo attuale scegliendo un certo vicino.
- Diversificazione: la diversificazione consiste nel cercare di individuare aree poco visitate dello spazio delle soluzioni, con lo scopo di individuare nuove aree promettenti su cui intensificare la ricerca. Per fare ciò ci sono varie strategie (come modificare l'aspirazione o la lunghezza della tabu list), ma si è deciso di passare dal 2-opt al 3-opt non appena si verifica una condizione di arresto. In questo modo viene espanso il vicinato ed aumenta la probabilità di trovare una soluzione migliore

Tempi

Questi sono i tempi ottenuti utilizzando la semplice ricerca locale (tempi misurati in laboratorio). L'algoritmo termina quando non sono più presenti vicini miglioranti.

Numero Nodi	LS random	LS circonferenza	LS 2 cluster	LS 5 cluster
10	0,0001	0,00015	0,0001	0,00015
20	0,00018	0,00022	0,0002	0,00021
30	0,0003	0,00035	0,0003	0,00035
40	0,0045	0,0005	0,00055	0,00055
50	0,0007	0,0009	0,0007	0,0007
60	0,001	0,0013	0,001	0,0011

70	0,002	0,003	0,002	0,002
80	0,002	0,0035	0,003	0,003
90	0,003	0,004	0,0025	0,004
100	0,004	0,005	0,004	0,005
1000	4,3	4,5	3,5	3,2

Tempi con Ricerca Locale e 2-opt

Di seguito invece vengono riportati i risultati della tabu search. La condizione d'arresto, non più l'assenza do vicini miglioranti, è cambiata. Le possibilità sono ora quindi due, che utilizzano due parametri. La prima consiste semplicemente nel fornire un numero massimo di iterazioni come parametro (100 in questo caso).
La seconda invece consiste nel fermarsi quando non ci sono più vicini "legali" disponibili. Un altro importante parametro è infatti la lunghezza della tabu-list. È stata scelta una lunghezza pari a 7, non troppo lunga da bloccare troppi vicini, e nemmeno troppo corta da consentire comunque di ciclare. I parametri usati sono sempre stati *tabuLength*=8 e *maxiter*=110 per poter fare un migliore confronto (casi a parte sono state le esecuzioni con 10 nodi, per le quali la tabu-length è stata abbassata in quanto troppo alta).

Numero Nodi	**TS random**	**TS circonferenza**	**TS 2 cluster**	**TS 5 cluster**
10	0,00015	0,00015	0,00015	0,00015
20	0,001	0,0015	0,0015	0,0015
30	0,002	0,002	0,002	0,002
40	0,003	0,004	0,003	0,003
50	0,004	0,003	0,004	0,004
60	0,004	0,003	0,004	0,004
70	0,0035	0,0035	0,0035	0,0035
80	0,0035	0,003	0,0035	0,004
90	0,0035	0,004	0,004	0,0045
100	0,005	0,0055	0,005	0,005
1000	0,41	0,413	0,4	0,401

Tempi con Tabu Search e 2-opt

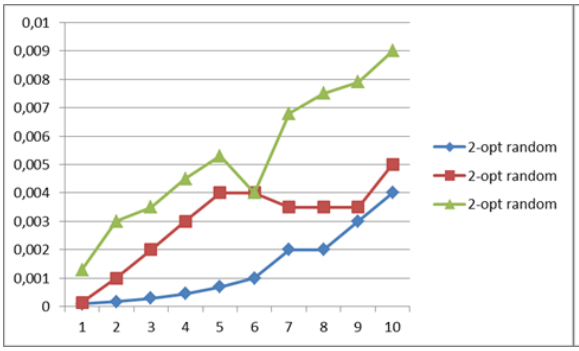
Per finire vengono elencati i tempi utilizzati con la tecnica dell'aspirazione.

Numero Nodi	**TSA random**	**TSA circonferenza**	**TSA 2 cluster**	**TSA 5 cluster**
10	0,0013	0,0014	0,0013	0,0015
20	0,003	0,003	0,003	0,003
30	0,0035	0,0035	0,0035	0,0035
40	0,0045	0,005	0,0045	0,005
50	0,0053	0,0055	0,0055	0,0055
60	0,004	0,0055	0,004	0,006
70	0,0068	0,0068	0,0068	0,0068
80	0,0075	0,0075	0,007	0,007
90	0,0079	0,0079	0,0077	0,0079
100	0,009	0,008	0,0085	0,009
1000	0,4	0,38	0,382	0,36

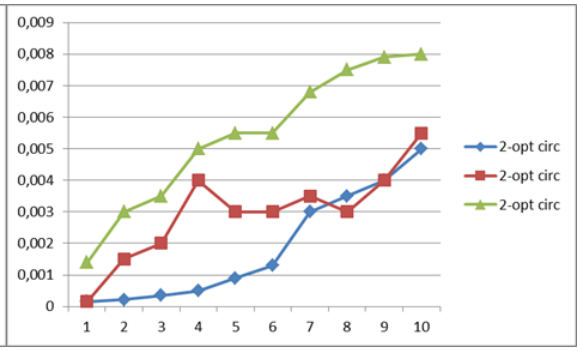
Tempi con Tabu-Search, aspirazione e 2-opt

È stata fatta una prova anche con un numero molto elevato di nodi per monitorare il comportamento del programma in queste condizioni. Usando la tabu-search (con eventuale aspirazione), il

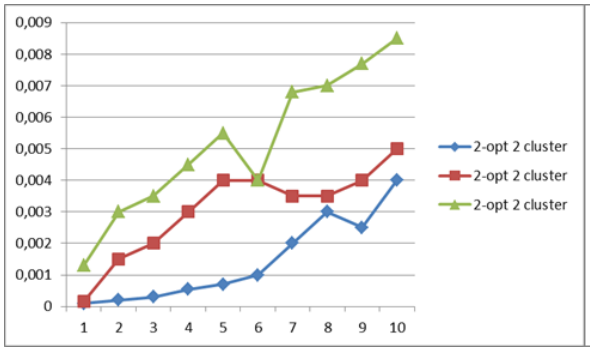
tempo ottenuto è di circa mezzo secondo. Un tempo molto più alto è stato impiegato per la ricerca locale. Questo perché il massimo numero di iterazioni non viene ovviamente usato come criterio di stop, e quindi la ricerca locale continuerà finché non verrà trovato più alcun vicino migliorante.
Sono stati eseguiti test anche con la soluzione iniziale alternativa. I risultati sono molto simili (leggermente più bassi), e sono dovuti al fatto che la soluzione iniziale era probabilmente migliore rispetto a quella casuale e sono stati necessari meno scambi.



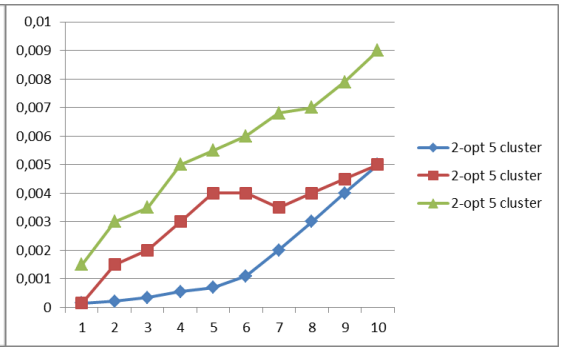
Tempi con random



Tempi con circonferenza



Tempi con 2 cluster



Tempi con 5 cluster

È in seguito stato misurato il tempo impiegato dal 3-opt rispetto al 2-opt con lo stesso numero di nodi (nuovamente Ricerca Locale, Tabu Search senza e con aspirazione).

Numero Nodi	LS random	LS circonferenza	LS 2 cluster	LS 5 cluster
10	0,0003	0,0002	0,0002	0,0001
20	0,0006	0,0005	0,0005	0,0006
30	0,0013	0,0016	0,0012	0,0016
40	0,0046	0,0046	0,0043	0,0039
50	0,0102	0,0106	0,0095	0,0099
60	0,0175	0,0162	0,0138	0,0190
70	0,0314	0,0335	0,0303	0,0326
80	0,0599	0,0539	0,0683	0,0648
90	0,0971	0,1067	0,0667	0,0957
100	0,1231	0,0980	0,1331	0,1206
500	70,2829	70,3812	61,3780	67,2477

Tempi con Ricerca Locale e 3-opt

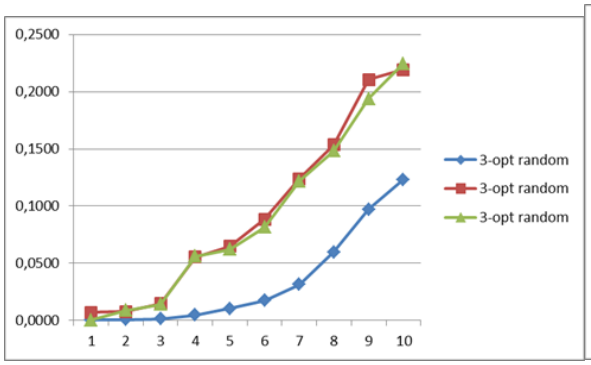
Numero Nodi	TS random	TS circonferenza	TS 2 cluster	TS 5 cluster
10	0,0069	0,0068	0,0058	0,0068
20	0,0078	0,0085	0,0089	0,0082

30	0,0147	0,0126	0,0157	0,0126
40	0,0554	0,0254	0,0255	0,0264
50	0,0648	0,0615	0,0602	0,0594
60	0,0885	0,0871	0,0866	0,0909
70	0,1238	0,0966	0,1199	0,1204
80	0,1536	0,1518	0,1588	0,1506
90	0,2107	0,1975	0,2154	0,1956
100	0,2192	0,2263	0,2243	0,2232
500	21,1665	21,1996	21,2452	21,2168

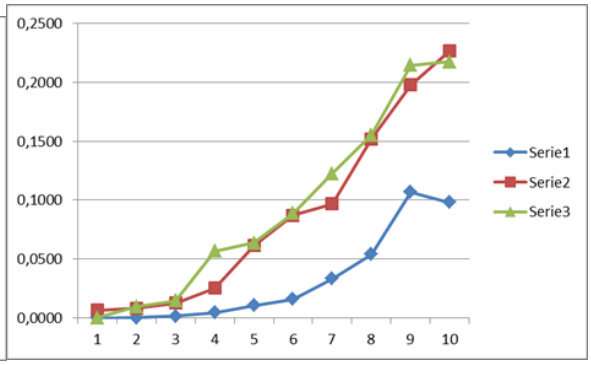
Tempi con Tabu-Search e 3-opt

Numero Nodi	TSA random	TSA circonferenza	TSA 2 cluster	TSA 5 cluster
10	0,0003	0,0003	0,0003	0,0003
20	0,0088	0,0096	0,0094	0,0095
30	0,0139	0,0146	0,0128	0,0115
40	0,0562	0,0567	0,0570	0,0494
50	0,0620	0,0636	0,0634	0,0643
60	0,0815	0,0888	0,0911	0,0904
70	0,1218	0,1225	0,1334	0,1197
80	0,1481	0,1553	0,1560	0,1538
90	0,1940	0,2143	0,1972	0,2135
100	0,2244	0,2171	0,2217	0,2274
500	23,5131	23,5631	23,2405	23,2569

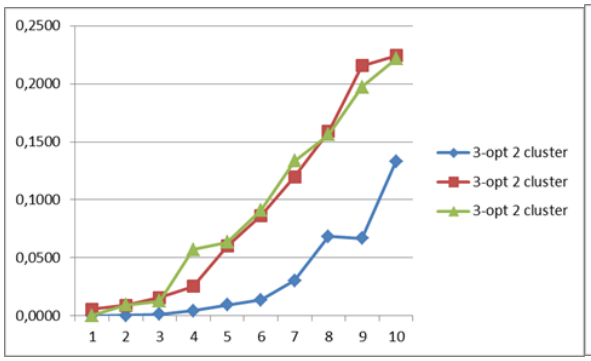
Tempi con Tabu-Search, aspirazione e 3-opt



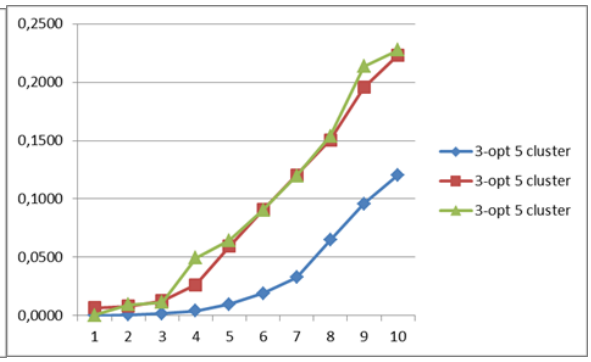
Tempi con random



Tempi con circonferenza



Tempi con 2 cluster



Tempi con 5 cluster

Si è infine misurato il tempo anche con la strategia di diversificazione.

Numero Nodi	LS random	LS circonferenza	LS 2 cluster	LS 5 cluster
10	0,0001	0,0001	0,0001	0,0001
20	0,0002	0,0002	0,0002	0,0002
30	0,0004	0,0004	0,0004	0,0004
40	0,0009	0,0007	0,0015	0,0006
50	0,0016	0,0011	0,0018	0,0010
60	0,0025	0,0020	0,0030	0,0021
70	0,0024	0,0032	0,0032	0,0024
80	0,0122	0,0038	0,0123	0,0058
90	0,0135	0,0047	0,0013	0,0127
100	0,0215	0,0065	0,0228	0,0202
500	8,3311	0,4787	4,1981	3,0942

Tempi con Ricerca Locale e diversificazione

Numero Nodi	TS random	TS circonferenza	TS 2 cluster	TS 5 cluster
10	DA FARE	DA FARE	DA FARE	DA FARE
20	DA FARE	DA FARE	DA FARE	DA FARE
30	DA FARE	DA FARE	DA FARE	DA FARE
40	DA FARE	DA FARE	DA FARE	DA FARE
50	DA FARE	DA FARE	DA FARE	DA FARE
60	DA FARE	DA FARE	DA FARE	DA FARE
70	DA FARE	DA FARE	DA FARE	DA FARE
80	DA FARE	DA FARE	DA FARE	DA FARE
90	DA FARE	DA FARE	DA FARE	DA FARE
100	DA FARE	DA FARE	DA FARE	DA FARE
500	DA FARE	DA FARE	DA FARE	DA FARE

Tempi con Tabu-Search e diversificazione

Numero Nodi	TSA random	TSA circonferenza	TSA 2 cluster	TSA 5 cluster
10	DA FARE	DA FARE	DA FARE	DA FARE
20	DA FARE	DA FARE	DA FARE	DA FARE
30	DA FARE	DA FARE	DA FARE	DA FARE
40	DA FARE	DA FARE	DA FARE	DA FARE
50	DA FARE	DA FARE	DA FARE	DA FARE
60	DA FARE	DA FARE	DA FARE	DA FARE
70	DA FARE	DA FARE	DA FARE	DA FARE
80	DA FARE	DA FARE	DA FARE	DA FARE
90	DA FARE	DA FARE	DA FARE	DA FARE
100	DA FARE	DA FARE	DA FARE	DA FARE
500	DA FARE	DA FARE	DA FARE	DA FARE

Tempi con Tabu-Search, aspirazione e diversificazione

Confronto tempi parte I e parte II

Verranno ora confrontate le tempistiche misurate dalla prima parte con quelle misurate in questa esercitazione (vengono presi in considerazioni solamente i tempi impiegati dal 2-opt, dato che gli altri restituiscono soluzioni molto vicine in un tempo però molto maggiore). Come si può vedere, le ricerche di questa parte di esercitazione utilizzano un tempo relativamente basso anche con un numero elevato di nodi. Questo dipende principalmente da due motivi.

Il primo consiste nel numero limitato di iterazioni nel caso della tabu search. Non dovendo più necessariamente trovare l'ottimo vero e proprio, ma solamente un valore "indicativo", non serve arrivare ad un numero altissimo di iterazioni.
La seconda consiste nel fatto che non è eseguito codice CPLEX per le ottimizzazioni. È usato del semplice codice C++ che esegue degli scambi e calcola delle somme in breve tempo.

Numero Nodi	Modello	2-opt	2-opt con tabu-list	2-opt con aspirazione
10	1	0,0001	0,00015	0,0013
20	1	0,00018	0,001	0,003
30	1	0,0003	0,002	0,0035
40	1	0,0045	0,003	0,0045
50	15	0,0007	0,004	0,0053
60	25	0,001	0,004	0,004
70	45	0,002	0,0035	0,0068
80	55	0,002	0,0035	0,0075
90	480	0,003	0,0035	0,0079
100	600	0,004	0,005	0,009
1000	-	4,3	4,5	3,5

Tempistiche con una disposizione dei punti casuale

Numero Nodi	Modello	2-opt	2-opt con tabu-list	2-opt con aspirazione
10	1	0,00015	0,00015	0,0014
20	2	0,00022	0,0015	0,003
30	2	0,00035	0,002	0,0035
40	2	0,0005	0,004	0,005
50	2	0,0009	0,003	0,0055
60	3	0,0013	0,003	0,0055
70	5	0,003	0,0035	0,0068
80	20	0,0035	0,003	0,0075
90	40	0,004	0,004	0,0079
100	55	0,005	0,0055	0,008
1000	-	4,5	3,5	3,2

Tempistiche con una disposizione dei punti casuale lungo una circonferenza

Numero Nodi	Modello	2-opt	2-opt con tabu-list	2-opt con aspirazione
10	1	0,0001	0,00015	0,0013
20	1	0,0002	0,0015	0,003
30	2	0,0003	0,002	0,0035
40	5	0,00055	0,003	0,0045
50	35	0,0007	0,004	0,0055
60	100	0,001	0,004	0,004
70	200	0,002	0,0035	0,0068
80	350	0,003	0,0035	0,007
90	-	0,0025	0,004	0,0077
100	-	0,004	0,005	0,0085
1000	-	3,5	3,2	0,41

Tempistiche con una disposizione dei punti casuale in 2 cluster

Numero Nodi	Modello	2-opt	2-opt con tabu-list	2-opt con aspirazione
10	1	0,00015	0,00015	0,0015
20	1	0,00021	0,0015	0,003
30	2	0,00035	0,002	0,0035
40	10	0,00055	0,003	0,005
50	55	0,0007	0,004	0,0055
60	350	0,0011	0,004	0,006
70	750	0,002	0,0035	0,0068
80	-	0,003	0,004	0,007
90	-	0,004	0,0045	0,0079
100	-	0,005	0,005	0,009
1000	-	3,2	0,41	0,413

Tempistiche con una disposizione dei punti casuale in 5 cluster

Confronto risultati parte I e parte II

Sono stati fatti infine dei test per poter confrontare i risultati ottenuti tramite le due realizzazioni. Per poterli eseguire sono stati generati dei file di testo contenenti le coordinate dei punti da usare, che sono stati quindi letti dai programmi e salvati.
Per i test si è deciso di utilizzare una disposizione casuale di 80 punti e 40 punti suddivisi in 2 cluster. La scelta deriva dal fatto che l'algoritmo esatto termina in un tempo accettabile con la prima disposizione di nodi, mentre nel secondo test, un numero alto di nodi avrebbe richiesto tempi abbastanza.
I parametri utilizzati per la tabu list sono stati calibrati prima dell'esecuzione del programma. Nel primo test vengono usati *tabuLength*=15 e *maxiter*=210. Nel secondo test invece abbiamo *tabuLength*=25 e *maxiter*=110.

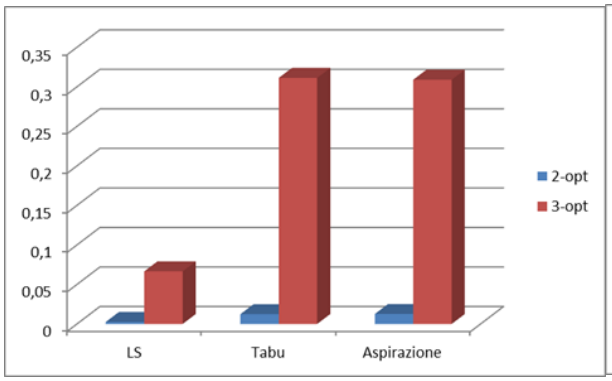
	Risultato	Tempo	Errore
Modello	**75,4979**	84,4011	0
2-opt LS	78,6892	0,00258493	4,055575606
2-opt Tabu	77,527	0,0123169	2,617281721
2-opt Aspirazione	76,763	0,012619	1,648059612
3-opt LS	78	0,066504	3,14207415
3-opt Tabu	76,1127	0,311268	0,814327286
3-opt Aspirazione	75,4979	0,308971	0
LS con diversificazione	77,1193	0,0100939	2,147609404
TS con diversificazione	**DA FARE**	**DA FARE**	**DA FARE**
TSA con diversificazione	**DA FARE**	**DA FARE**	**DA FARE**

Risultati con una disposizione di 80 punti casuale

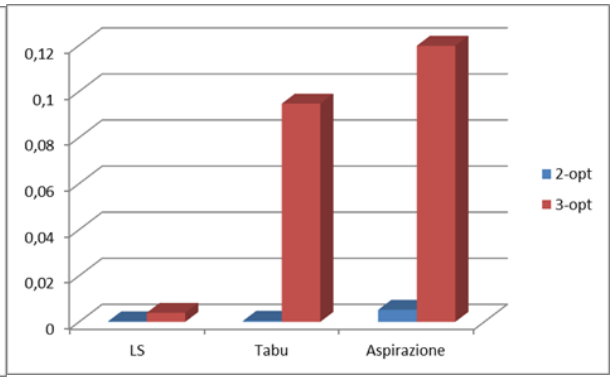
	Risultato	Tempo	Errore
Modello	**59,8062**	8,32829	0
2-opt LS	63,6385	0,000218153	6,021983548
2-opt Tabu	60,3453	0,000463009	0,893358721
2-opt Aspirazione	59,8063	0,00520706	0,000167206
3-opt LS	61,1608	0,00392199	2,26498256
3-opt Tabu	60,58	0,0946829	1,293845789
3-opt Aspirazione	59,8063	0,119809	0,000167207
LS con diversificazione	59,8498	0,00107789	0,072902141
TS con diversificazione	**66,1674**	**0,000904799**	**10,63635543**
TSA con diversificazione	**DA FARE**	**DA FARE**	**DA FARE**

Risultati con una disposizione di 40 punti casuale in 2 cluster

Per ogni test è stato salvato il risultato, il tempo impiegato (in secondi) e l'errore relativo rispetto al valore trovato col modello della prima esercitazione. Come ci si poteva aspettare, il risultato peggiore è stato ottenuto con la ricerca locale, in quanto fermandomi quando non trovo più soluzioni migliori, non è detto che possa raggiungere l'eventuale ottimo. Risultati molto migliori sono stati ottenuti invece con la tabu-search e con l'uso dell'aspirazione. I seguenti grafici mostrano invece l'aumento del tempo impiegato rispettivamente con 2-opt e con 3-opt.



80 punti casuali



40 punti casuali in 2 cluster

Come si può vedere, il 3-opt ritorna quasi gli stessi risultati, però arriva ad aumento del tempo addirittura del 2200% nell'ultimo caso, rendendo quindi inutile l'uso di tale strategia.