POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica

Tesi di Laurea Magistrale

Problema del commesso viaggiatore: un punto di vista basato sul feedback positivo

Relatori
prof. Luigi Preziosi
prof. Marco Scianna
firma dei relatori

. . . . . .
. . . . . .

Candidato
Luca Campanello

firma del candidato

. . . . . .

Anno Accademico 2020-2021

Ringraziamenti

Vorrei ringraziare i miei relatori Luigi Preziosi e Marco Scianna per il continuo e prezioso supporto nella stesura di questo testo. La loro esperienza e i loro suggerimenti sono stati fondamentali in tutte le fasi del lavoro, a partire dalla ricerca del materiale e dalla struttura generale della tesi, fino allo svolgimento delle simulazioni e alla presentazione formale dei risultati. Vorrei ringraziare inoltre i miei genitori per avermi offerto questa opportunità e successivamente sostenuto con costanza e interesse lungo tutto il cammino, sapendomi infondere fiducia nei miei mezzi e aiutandomi a dare il meglio di me. Vorrei ringraziare infine tutte le persone che hanno avuto un ruolo in questo percorso, attraverso il confronto con le quali ho sperimentato occasioni di crescita a livello accademico e umano.

Indice

Elenco delle figure

Sommario

In questo testo verranno presentate strategie meta-euristiche applicate a un noto problema di ottimizzazione stocastica: il problema del commesso viaggiatore. La dimensione dello spazio di ricerca rende difficile l'adozione di tecniche più analitiche, orientando il percorso verso metodi intuitivi che agiscano attraverso il rilevamento di soluzioni sub-ottime via via più accurate. Alla base degli algoritmi vi è l'interazione tra agenti semplici, ovvero dotati di un numero limitato di operazioni e di scelte, i quali procedono per tentativi che vengono progressivamente raffinati tramite le informazioni che ciascun agente riceve dagli altri. Tale processo di comunicazione viene integrato da un'euristica base che permette di indirizzare la ricerca nelle prime fasi. Infine, una strategia di calcolo distribuito è utile nell'ostacolare il processo dalla convergenza a un ottimo locale.

I primi cinque capitoli prendono spunto dal paper "Positive Feedback as a Search Strategy" ad opera di M.Dorigo, V.Maniezzo, A.Colorni (Dorigo et al. [1991]) nei modelli utilizzati e nel tipo di analisi svolte. In particolare, nel capitolo 1 viene definito il problema e vengono presentati alcuni possibili approcci a esso, tra cui quello ad agenti che è stato effettivamente utilizzato per risolverlo. Nel capitolo 2 vengono descritte nel dettaglio le dinamiche del modello e quindi il funzionamento dell'algoritmo. Nel capitolo 3 sono riportati alcuni risultati relativi a una topologia regolare di griglia 4×4. Nel capitolo 4 sono presentati risultati relativi a una topologia irregolare, il problema Oliver30. Nel capitolo 5 viene presentata una variante dell'algoritmo classico con i relativi risultati di convergenza. Nel capitolo 6 vengono introdotte nuove varianti dell'algoritmo e diverse analisi sui modelli che ne risultano. Nel capitolo 7 l'algoritmo viene applicato ad altre topologie. Nel capitolo 8 sono presentate le conclusioni. In appendice sono riportati i codici utilizzati.

Capitolo 1

Introduzione

Uno degli approcci utilizzabili per affrontare problemi di ottimizzazione prevede la compresenza di molteplici agenti: ciascuno di questi compie un proprio tentativo di risoluzione e comunica poi agli altri la "bontà" del risultato trovato. In generale, un problema di ottimizzazione consiste nell'individuare

$$\underset{x \in \mathcal{S}}{\operatorname{argmin}} f(x), \tag{1.1}$$

dove f è una funzione data e $\mathcal{S}$ è un insieme dato. Ogni agente esegue una sequenza di operazioni, ciascuna delle quali selezionando tra un numero finito di possibili alternative, per ottenere un valore che si avvicini il più possibile a quello in formula (1.1). Tramite un sistema di feedback, poi, tale agente "consiglia" di più o di meno quella sequenza di scelte agli altri, in base a quanto si è rivelata efficace. Di conseguenza, ciascuno degli agenti può servirsi dei feedback di quelli che hanno già eseguito i propri tentativi per le scelte che deve compiere. Inoltre, la procedura viene ripetuta un grande numero di volte, in modo che si abbiano molti feedback a cui fare riferimento.

L'approccio descritto è stato applicato a un problema di routing molto noto in letteratura: quello del commesso viaggiatore, ovvero il "travel salesman problem" (TSP) (Grötschel et al. [1985]). Nella sua formulazione tradizionale, il dominio spaziale è un grafo bidimensionale $\mathcal{G}$ in cui n punti $\{c_i \mid i \in \{1, ..., n\}\}$, ciascuno di coordinate fisse (x_i, y_i), sono interconnessi tra loro da segmenti (Fig. (1.1a)). Sia inoltre $d_{i,j}$ la lunghezza euclidea del segmento $s_{i,j}$, che congiunge i nodi c_i e c_j. Con la notazione "cammino lecito" o "ciclo" sarà indicato quindi ogni percorso chiuso che passa una e una sola volta in tutti gli n nodi (a partire da un nodo arbitrariamente scelto). Ognuno dei possibili cicli è dunque identificato dalla sequenza ordinata di nodi percorsi in successione: $\mathbf{c} = (c_k)_{k \in A(i=1,...,n)}$, dove A indica una permutazione degli indici $1, ..., n$. La lunghezza di un generico ciclo $\mathbf{c}$, identificata da $L_{\mathbf{c}}$, è data dalla somma delle lunghezze dei segmenti che lo compongono (Fig. (1.1b) e (1.1c)). Per esempio, per un grafo con $n = 4$ nodi, $\mathbf{c} = (c_3, c_2, c_1, c_4)$ indica il ciclo che parte dal nodo c_3, percorre nell'ordine i nodi c_2, c_1 e c_4 e si ricongiunge infine al nodo c_3; la lunghezza di $\mathbf{c}$ è data da $L_{\mathbf{c}} = d_{3,2} + d_{2,1} + d_{1,4} + d_{4,3}$.

In quest'ottica, il problema del commesso viaggiatore consiste nell'individuare il ciclo di lunghezza minima data una specifica distribuzione di nodi (uno specifico grafo). Il ciclo più breve sarà denotato come quello "ottimo" o "minimo", mentre "l'ottimo" o "il minimo" sarà la lunghezza corrispondente. L'insieme di tutti i cicli sarà indicato con $\mathcal{C}$ e consiste nell'insieme di tutte le possibili permutazioni dell'ordine dei nodi $\{c_i \mid i \in \{1, ..., n\}\}$. In Fig.(1.1) è riportato un esempio di grafo con il relativo ciclo minimo.

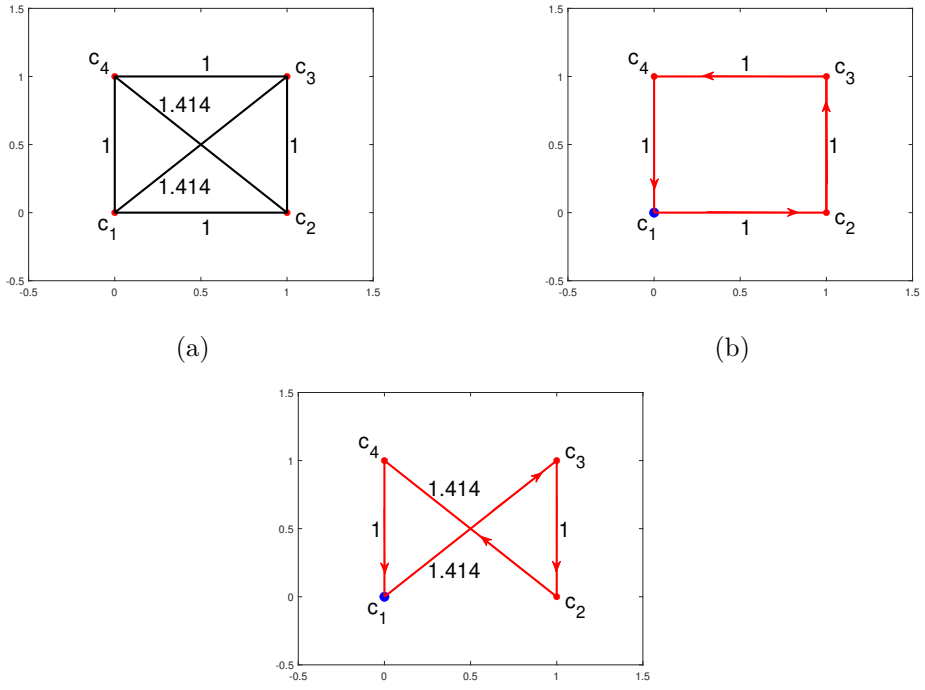


Figura 1.1: Grafo formato da $n = 4$ nodi. (a) I segmenti neri congiungono ogni coppia di nodi. (b) Ciclo $\mathbf{c}' = (c_1, c_2, c_3, c_4)$, la cui lunghezza è pari a $L_{\mathbf{c}'} = 4$. (c) Ciclo $\mathbf{c}'' = (c_1, c_3, c_2, c_4)$, di lunghezza $L_{\mathbf{c}''} = 4.828$. Il ciclo $\mathbf{c}'$ rappresenta anche il cammino minimo.

Un primo approccio intuitivo (non ad agenti) al problema potrebbe essere quello della ricerca esaustiva, ovvero calcolare la lunghezza di tutti i possibili cicli; tuttavia, il costo computazionale di tale strategia non sarebbe sostenibile per n molto elevato, poiché occorrerebbe calcolare $n!$ cammini. Un altro metodo ancora intuitivo, ma leggermente più elaborato (anch'esso non ad agenti) potrebbe essere quello di scegliere di volta in volta il nodo più vicino, ovvero:

- selezionare un nodo casuale c_k come partenza;
- individuare il sottoinsieme dei segmenti che hanno c_k come estremo;

- selezionare il segmento $s_{j,k} \mid d_{j,k} = \min_{i \neq k} d_{i,k}$;
- dal nodo c_j di destinazione, ripetere il processo con l'insieme dei segmenti che hanno c_j come estremo, ma escludendovi il segmento $s_{j,k}$, poiché c_k è stato già visitato;
- iterare il processo fino ad avere visitato tutti i nodi, escludendo ogni volta nella scelta quelli già visitati.

Il limite di questo metodo è che viene esclusa dalla ricerca la maggior parte di tutti i possibili percorsi: in particolare, potrebbe essere impossibile rilevare il cammino ottimo, nel caso questo implichi la scelta, ad una determinata iterazione, di un segmento consentito non di lunghezza minima.

La strategia che si è invece qui adottata è un metodo ad agenti: ciascuno di questi parte da un nodo e percorre un ciclo, in una ricerca parallela dell'ottimo. Il sistema che verrà descritto si ispira al comportamento reale delle colonie di formiche (Deneubourg et al. [1983],Deneubourg and Goss [1989],Goss et al. [1990]): un ruolo fondamentale è rappresentato da una sostanza che queste rilasciano continuamente durante il loro cammino, il ferormone. Quando una formica si muove e deve selezionare quindi una direzione, rileva la quantità di ferormone presente sui percorsi disponibili e tende a dirigersi verso il tratto in cui la traccia di questa sostanza è più intensa: le formiche hanno infatti una vista molto poco acuta e utilizzano questa strategia per riconoscere i cammini migliori. Ogni volta che un segmento viene percorso, aumenta la probabilità che esso venga selezionato successivamente da altri agenti, poiché l'intensità di ferormone presente su un segmento è chiaramente conseguenza diretta di tutti gli agenti/formiche che lo hanno percorso. Si tratta di un sistema autocatalitico (Dorigo [1992]): più spesso un cammino viene selezionato, più intensa sarà la traccia di sostanza su di esso e più frequentemente, di conseguenza, sarà nuovamente scelto in futuro. In questo modo è stata implementata la strategia di feedback sopra descritta. Si noti inoltre che la compresenza di molteplici agenti è fondamentale: se, infatti, ve ne fosse uno solo, questo ripercorrerebbe iterativamente gli stessi percorsi, non riuscendo a esplorare tutto lo spazio in modo efficace e provocando una convergenza a un cammino sub-ottimo. Gli algoritmi implementati prendono spunto da questo modello reale, con alcune necessarie modifiche e semplificazioni: il tempo è discretizzato, le formiche hanno una vista che permette loro di valutare la distanza dei nodi circostanti e inoltre ogni agente sa quali nodi ha già visitato, altrimenti non sarebbe possibile riprodurre il vincolo per cui ogni nodo deve essere visitato una volta sola.

Capitolo 2

Descrizione dell'algoritmo

In questo capitolo viene definito nel dettaglio il funzionamento dell'algoritmo. Dato un generico grafo $\mathcal{G}$ costituito da n nodi, scopo dell'algoritmo è individuare

$$\mathbf{c}^* := \underset{\mathbf{c}\in\mathcal{C}}{\operatorname{argmin}} L_{\mathbf{c}}, \tag{2.1}$$

dove $\mathcal{C}$ è l'insieme di tutti i possibili cicli su $\mathcal{G}$.

Innanzitutto, il modello prevede la presenza di M agenti $\{a_m \mid m \in \{1, ..., M\}\}$, ciascuno dei quali percorre un proprio ciclo su $\mathcal{G}$. Il vettore $\mathbf{b}(t) \in \mathbb{N}_0^n$ ha come generica componente $b_k(t)$ il numero di agenti che al tempo t sono presenti sul nodo c_k. Quindi, $\sum_{k=1}^{n} b_k(t) = M,\ \forall t \geq 0$. Sarà identificata con $q_{i,j}(t) \in \mathbb{R}_0^+$ la quantità di ferormone presente ad un generico istante t su un generico segmento $s_{i,j}$. L'intensità di ferormone sul generico segmento $s_{i,j}$ al tempo t, denotata con $\tau_{i,j}(t) := \frac{q_{i,j}(t)}{d_{i,j}}$, è definita come il rapporto tra la quantità di ferormone sul segmento e la lunghezza del segmento stesso. Un generico agente a_m che si trova sul nodo c_i all'istante t sceglierà la sua destinazione successiva c_j con la seguente probabilità:

$$p_{i,j}^m(t) = \begin{cases} \dfrac{[\tau_{i,j}(t)]^\alpha [\eta_{i,j}]^\beta}{\sum_{h\in\mathcal{G}^m(t)} [\tau_{i,h}(t)]^\alpha [\eta_{i,h}]^\beta} & \text{se } j \in \mathcal{G}^m(t) \\ 0 & \text{se } j \notin \mathcal{G}^m(t). \end{cases} \tag{2.2}$$

Nell'equazione (2.2):

- con $G^m(t)$ è indicato il sottoinsieme di nodi che l'agente a_m non ha ancora visitato nel suo ciclo, e che quindi sono possibili destinazioni per a_m all'istante t;
- $\eta_{i,j}$ indica la visibilità tra i nodi c_i e c_j, che è definita come l'inverso della loro lunghezza: $\eta_{i,j} := d_{i,j}^{-1}$;

- α e β sono due parametri strettamente positivi che quantificano la relativa importanza nella scelta del percorso da parte degli agenti, rispettivamente, della presenza di feroromone e della visibilità.

La quantità di ferormone su ogni segmento del grafo $\mathcal{G}$ varia, a partire dall'istante iniziale t_0, ad ogni $q \in \mathbb{N}$ step. In quest'ottica, per ogni segmento $s_{i,j}$, si può scrivere:

$$\tau_{i,j}(t_0 + (k+1)q)) = \rho\tau_{i,j}(t_0 + kq) + \mathcal{I}(\tau_{i,j}), \tag{2.3}$$

dove $k \in \mathbb{N}$, ρ indica la frazione di ferormone che rimane (ovvero non evapora) in un intervallo di tempo di lunghezza q e $\mathcal{I}(\tau_{i,j})$ è una funzione che indica l'intensità di ferormone rilasciata dagli agenti che hanno percorso il segmento $s_{i,j}$ nel lasso di step considerato. Le diverse scelte di q, ovvero della frequenza di aggiornamento del ferormone sui segmenti che connettono i nodi, definiscono diversi algoritmi. In particolare, se $q = 1$ l'aggiornamento avviene ad ogni step: questo caraterizza gli algoritmi di tipo "ant-density" e "ant-quantity", che a loro volta si differenziano per la quantità di ferormone che viene emessa dagli agenti. In particolare, nel modello "ant-density", il rilascio avviene a intensità fissa: ogni volta che un agente percorre un generico segmento $s_{i,j}$, vi rilascia una quantità di ferormone direttamente proporzionale alla lunghezza $d_{i,j}$ del segmento stesso; nel modello "ant-quantity", invece, è la quantità di ferormone emessa su ogni segmento a essere fissa, il che produce un'intensità maggiore sui segmenti più brevi. Se invece $q = n$, l'aggiornamento avviene ad ogni fine ciclo percorso dagli agenti: a riguardo è infatti facile osservare come ogni agente, per toccare una e una sola volta tutti gli n nodi, necessita di n step. Questa sarà la scelta su cui verrà costruito l'algoritmo. La funzione $\mathcal{I}(\tau_{i,j})$ introdotta precedentemente è definita come segue:

$$\mathcal{I}(\tau_{i,j}) := \sum_{\substack{m=1 \\ m \in \mathcal{S}_{i,j}(t_0+kq, t_0+(k+1)q)}}^{M} \Delta\tau_{i,j}^{m}, \tag{2.4}$$

dove $\mathcal{S}_{i,j}(t_0+kq, t_0+(k+1)q)$ indica l'insieme degli agenti che hanno percorso il segmento $s_{i,j}$ nell'intervallo di tempo di riferimento $(t_0 + kq, t_0 + (k+1)q)$, mentre $\Delta\tau_{i,j}^{m}$ è la quantità di ferormone rilasciata da ognuno di essi sul segmento $s_{i,j}$, sempre nell'intervallo di riferimento. Si è qui optato per il rilascio del ferormone con quantità fissa per ogni ciclo, dunque si ha:

$$\Delta\tau_{i,j}^{m} = Q/L_{\mathbf{c}_m}, \tag{2.5}$$

dove Q è un valore predefinito e $L_{\mathbf{c}_m}$ è la lunghezza del ciclo percorso dall'agente a_m nel lasso di tempo $(t_0 + kq, t_0 + (k+1)q)$. Il modello presentato, con le scelte di $q = n$ e del rilascio a quantità fissa, prende il nome di "ant-cycle"; il vantaggio di questo approccio è che viene distribuita un'intensità di ferormone più elevata sui cicli complessivamente più brevi, il che li rende preferibili, facilitando quindi la convergenza al ciclo ottimo.

L'algoritmo è inizializzato dalle posizioni degli agenti e dalla quantità di ferormone presente sul grafo. Sono poi previste $k = 1, ..., NI_{max}$ iterazioni, ognuna corrispondente ad un ciclo intero percorso da tutti gli agenti e ognuna costituita da n step, ciascuno dei

quali corrisponde a sua volta a un segmento percorso dagli agenti, secondo la legge del moto (2.2). Al termine di ogni iterazione k, viene aggiornata l'intensità di ferormone sul grafo secondo la legge (2.3) e vengono aggiornati il ciclo minimo $\mathbf{c}^*$ e la sua lunghezza $L^*_{\mathbf{c}}$ (se necessario, ovvero se almeno un agente ha compiuto un ciclo di lunghezza inferiore a quello precedementemente salvato). Al termine delle NI_{max} iterazioni, $\mathbf{c}^*$ sarà il ciclo minimo trovato dagli agenti; è possibile inoltre impostare una condizione di arresto preliminare, come il percorrimento del medesimo ciclo da parte di tutti gli agenti in una generica iterazione, oppure il caso in cui la lunghezza minima del ciclo trovato sia inferiore ad una certa soglia.

L'algoritmo presentato ha dunque lo scopo di individuare il ciclo sul grafo $\mathcal{G}$ di lunghezza minima; a questo scopo, è utile introdurre una relazione di similitudine tra cicli, che si fondi proprio sul confronto tra le lunghezze dei cicli stessi. Innanzitutto, come detto, un ciclo $\mathbf{c}$ corrisponde a un ordine di percorrimento dei nodi $\{c_i \mid i \in \{1, ..., n\}\}$; si considerino come esempio i seguenti due cicli:

- $\mathbf{c}^1 = (c_1, c_2, c_3, ..., c_n)$;
- $\mathbf{c}^4 = (c_4, c_5, c_6, ..., c_n, c_1, c_2, c_3)$.

$\mathbf{c}^1$ e $\mathbf{c}^4$ appaiono diversi, tuttavia differiscono unicamente per il nodo selezionato come partenza: il cammino chiuso di fatto è lo stesso, dunque si ha $L_{\mathbf{c}^1} = L_{\mathbf{c}^4}$, perciò è utile definirli simili e scrivere $\mathbf{c}^1 \sim \mathbf{c}^4$. Estendendo questo ragionamento, si consideri il ciclo $\mathbf{c}^k = (c_k, ..., c_n, c_1, ..., c_{k-1})$, con $k \in \{1, ..., n\}$: i cammini chiusi di $\mathbf{c}^1$ e $\mathbf{c}^k$ sono gli stessi $\forall\, k$ e anche le loro lunghezze, quindi ha senso dire che $\mathbf{c}^1 \sim \mathbf{c}^k\ \forall\, k \in \{1, ..., n\}$. Analogamente, il verso di percorrenza di un determinato ciclo non ha effetto sulla sua lunghezza: è conveniente considerare il ciclo $\mathbf{c}^{-1} = (c_n, c_{n-1}, ..., c_2, c_1)$ simile a $\mathbf{c}^1$. Seguendo questa linea, si ha anche $\mathbf{c}^4 \sim \mathbf{c}^{-1}$, il che conferma che si tratta di una relazione transitiva.

La definizione di similitudine introdotta serve a presentare un pattern che sarà utile nello studio della convergenza dell'algoritmo, ovvero il cosiddetto "comportamento di percorso unico": si tratta della situazione nella quale tutti gli agenti, da un'iterazione k_0 in poi, percorrono il medesimo ciclo o cicli simili. Da un punto di vista matematico, siano $c_{p_{i,\mathbf{c}}}$ e $c_{s_{i,\mathbf{c}}}$ i due nodi che, rispettivamente, precedono e seguono il nodo c_i nel ciclo $\mathbf{c}$, $\forall\, i \in \{1, ..., n\}$. Si ha allora comportamento di percorso unico quando è verificata la seguente condizione:

$$\exists\, \mathbf{c} \in \mathcal{C} \mid\ \forall\, i \in \{1, ..., n\},\ \forall\, m \in \{1, ..., M\},\ \lim_{t \to +\infty} p^m_{i,j}(t) = 0\ \forall\, j \neq p_{i,\mathbf{c}}, s_{i,\mathbf{c}}, \tag{2.6}$$

dove $p^m_{i,j}(t)$ è la probabilità di transizione definita in Eq.(2.2). La definizione introdotta vale $\forall\, M \geq 1$: per $M = 1$, in particolare, si ha percorso unico quando l'unico agente a_1 percorre cicli simili (o lo stesso ciclo) al variare delle iterazioni, a partire da un'iterazione k_0. Si noti che, in quest'ultimo caso, ogni ciclo $\mathbf{c}$ è simile solo a se stesso e a $\mathbf{c}^{-1}$, ovvero $\mathbf{c}$ percorso in senso inverso, poiché l'unico agente a_1 parte sempre dallo stesso nodo. Nelle simulazioni, non essendo possibile confrontare i cicli $\forall\, k \geq k_0$ né verificare la condizione (2.6), si è supposto di avere percorso unico quando il pattern viene osservato per 100

iterazioni consecutive. E' evidente che, quando si ha comportamento di percorso unico, non si avrebbe alcun vantaggio nel proseguire la ricerca, poiché tutti gli M agenti si sono stabiliti su un ciclo specifico (che può essere ottimo o meno, ma in ogni caso non ne saranno individuati altri nelle iterazioni successive). Quando viene individuato percorso unico, l'algoritmo si arresta.

Capitolo 3

Risultati per la topologia di griglia

In questo capitolo e nei successivi sono state effettuate delle simulazioni tramite il software Matlab con lo scopo di testare il funzionamento dell'algoritmo descritto su topologie predefinite. Il grafo di riferimento, che sarà da ora denominato $\mathcal{G}_4$, è costituito da 16 nodi uniformemente distribuiti nell'area $[1,4] \times [1,4]$ dello spazio $\mathbb{R}^2$. Di tale griglia conosciamo, per via analitica, i cicli minimi e la loro lunghezza, che è pari a 16. $\mathcal{G}_4$ è rappresentato in Fig.(3.1) e due dei suoi cicli minimi in Fig.(3.2).

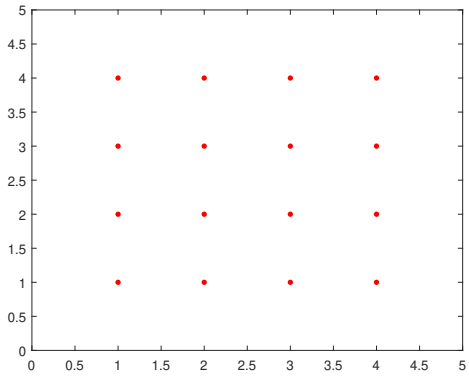
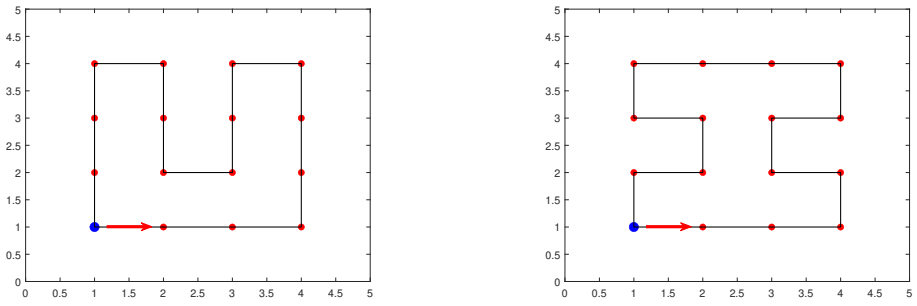


Figura 3.1: Griglia 4×4, regolare

Per le simulazioni, l'algoritmo è stato inizializzato come segue:

- il numero M di agenti è pari a quello n dei nodi;
- gli agenti vengono inizialmente distribuiti in modo uniforme, quindi uno per nodo;



(a) (b)

Figura 3.2: Due cicli minimi della griglia 4×4. Altri cicli minimi sono simili a questi, con la notazione introdotta nel capitolo 2, oppure si possono ottenere da essi tramite rotazione di un angolo multiplo di $\frac{\pi}{2}$.

- la quantità Q di ferormone introdotta in Eq.(2.5) da rilasciare su ogni ciclo è preimpostata a 100;
- l'intensità iniziale di ferormone sui segmenti di $\mathcal{G}_4$ è $\tau_{i,j}(0) = 10,\ \forall(i,j)$;
- i valori standard dei parametri sono $\{\alpha = 1, \beta = 1, \rho = 0.7\}$;
- ogni simulazione prevede $NI_{max} = 5000$ iterazioni, salvo raggiungimento delle condizioni di arresto descritte nel capitolo 2.

3.1 Convergenza al minimo

Saranno ora analizzate le variazioni del comportamento degli agenti nella ricerca del percorso minimo in funzione della variazione dei parametri α e β dell'Eq.(2.2). In particolare, sarà esaminato se, per ogni coppia di valori dei parametri presi nell'intervallo $\alpha \times \beta = [0,5] \times [0,5]$, il gruppo di agenti riesca a individuare il percorso minimo reale ed, eventualmente, in quante iterazioni. I risultati sono illustrati in Fig.(3.3).

Il primo risultato evidente dalla Fig.(3.3) è la convergenza immediata dell'algoritmo per $\beta \geq 3$, che si verifica per ogni valore di $\alpha \in [0,5]$. Questo fenomeno si può interpretare analizzando i ruoli di α e β nell'Eq.(2.2). Da (2.2) si ricava infatti che, per un generico agente a_m che si trova sul generico nodo c_i al tempo t e non ha ancora visitato i generici nodi c_j e c_k, con $j, k \neq i$:

$$r_i^{j,k}(t) := \frac{p_{i,j}^m(t)}{p_{i,k}^m(t)} = \frac{[\tau_{i,j}(t)]^\alpha\ [\eta_{i,j}]^\beta}{[\tau_{i,k}(t)]^\alpha\ [\eta_{i,k}]^\beta} = \left[\frac{\tau_{i,j}(t)}{\tau_{i,k}(t)}\right]^\alpha \left[\frac{\eta_{i,j}}{\eta_{i,k}}\right]^\beta, \tag{3.1}$$

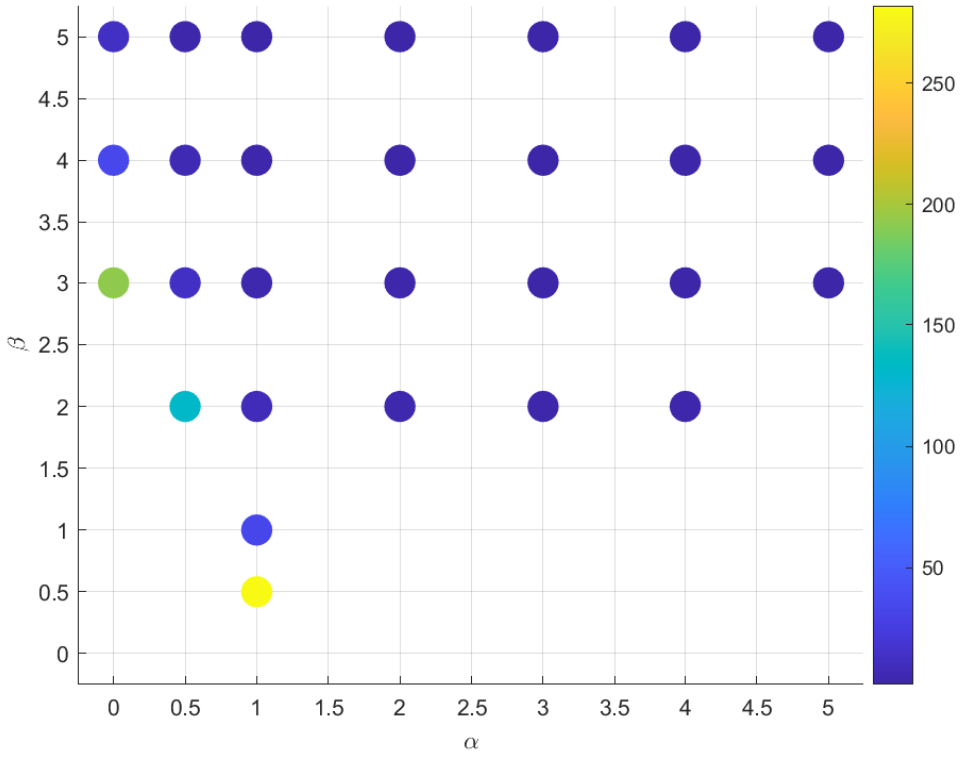


Figura 3.3: Numero di iterazioni necessarie per individuare il ciclo di lunghezza minima nella griglia 4×4 al variare di $\{\alpha, \beta\}$ in $[0,5] \times [0,5]$. I risultati sono mediati su 20 trials. Per le combinazioni per le quali il minimo non è stato individuato con efficacia (meno del 75% dei trials) entro 1000 iterazioni, non è stato rappresentato alcun valore. Si nota che l'algoritmo converge sempre per $\beta \geq 3$ e lo fa rapidamente, a prescindere dal valore di α; per $\beta < 3$, invece, la ricerca è più efficace per α vicino a 1.

dove $r_i^{j,k}(t)$ è il rapporto di preferibilità tra i nodi c_j e c_k partendo dal nodo c_i al tempo t, per il quale:

$$\begin{cases} r_i^{j,k}(t) > 1 \implies c_j \text{ è preferibile a } c_k \text{ partendo da } c_i \text{ in } t \\ r_i^{j,k} = 1 \implies c_j \text{ e } c_k \text{ sono ugualmente preferibili partendo da } c_i \text{ in } t \\ r_i^{j,k} < 1 \implies c_k \text{ è preferibile a } c_j \text{ partendo da } c_i \text{ in } t. \end{cases}$$

Se sia il criterio di visibilità che il criterio del ferormone favoriscono lo stesso nodo nella scelta, per esempio:

$$\begin{cases} \tau_{i,j}(t) > \tau_{i,k}(t) \\ \eta_{i,j} > \eta_{i,k}, \end{cases}$$

il rapporto di preferibilità è chiaro (in questo caso $r_i^{j,k}(t) > 1$) a prescindere dai valori di α e β. Al contrario, dall'Eq.(3.1) si deduce che, se i due criteri favoriscono nodi diversi, i valori di α e β sono decisivi nella scelta: in particolare, $\alpha \gg \beta$ rende il criterio del ferormone predominante, viceversa per $\beta \gg \alpha$. Questo ragionamento spiega il motivo per cui, per $\beta > \alpha$, gli agenti presentano una forte tendenza a selezionare di volta in volta i nodi più vicini a quello corrente. Nel caso particolare di $\mathcal{G}_4$ (e delle griglie regolari più in generale), si tratta dei nodi adiacenti verticalmente e orizzontalmente, gli unici per i quali la distanza è pari a 1. Osservando le Fig. (3.2a) e (3.2b), si nota che i cicli minimi sono composti unicamente da segmenti che connettono proprio nodi adiacenti orizzontalmente o verticalmente tra loro: dunque, nel modello analizzato, $\beta > \alpha$ è spesso sufficiente per avere convergenza al minimo. In realtà, la Fig.(3.3) mostra che la ricerca converge velocemente anche per $\alpha \geq \beta$, quando comunque $\beta \geq 3$: ciò accade perché nelle prime fasi della ricerca β è più importante di α, in quanto il ferormone distribuito sui segmenti è ancora poco e non significativo; di conseguenza, α elevato non fa altro che forzare la ricerca sui cicli individuati per primi, ovvero quelli che connnettono nodi adiacenti tra loro. Per $\beta = 0$, al contrario, l'algoritmo non converge per alcun valore di $\alpha \in [0,5]$: il solo criterio del ferormone è inefficace, perché manca una guida nelle fasi iniziali. Per $\beta \in [0.5,2]$, invece, la convergenza dipende dal valore di α: in particolare, la ricerca appare inefficace per valori agli estremi dell'intervallo [0,5] e ottimale per α vicino a 1. Infatti, α deve avere un valore sufficientemente alto perché il ruolo del ferormone non sia azzerato, ma sufficientemente basso per evitare che la ricerca si stabilizzi su pochi segmenti (quelli con più intensità di ferormone nelle fasi iniziali) e non riesca a esplorare adeguatmente tutto lo spazio.

Riassumendo:

- per $\beta \geq 3$ si ha sempre convergenza e α non ha impatto;
- per $\beta = 0$ non si ha mai convergenza e α non ha impatto;
- per $\beta \in [0.5,2]$ la convergenza dipende dal valore di α;
- fissando α, la convergenza dipende in ogni caso fortemente dal valore di β.

Se ne deduce che, per $\mathcal{G}_4$, il criterio del ferormone è secondario: un valore di β elevato, che conferisce rilevanza al criterio della visibilità, è sufficiente a rendere efficace la ricerca.

3.2 Analisi del percorso unico

Dopo avere analizzato la capacità dell'algoritmo di trovare il ciclo di lunghezza minima al variare di α e β, si cercherà ora di capire se tale percorso venga individuato da tutti gli M agenti o da un loro sottoinsieme. In altre parole, sarà analizzato se, al variare della coppia di parametri $\{\alpha, \beta\}$, il sistema di agenti si stabilizzi o meno nella condizione di percorso unico, descritta nel capitolo 2. Si noti che è anche possibile che la dinamica presenti percorso unico, ma che tale percorso non sia uno dei cicli minimi. La Fig.(3.4) mostra i sottoinsiemi di valori di $\{\alpha, \beta\} \in [0,5] \times [0,5]$ per i quali la ricerca converge a percorso unico e se tale percorso sia il minimo.

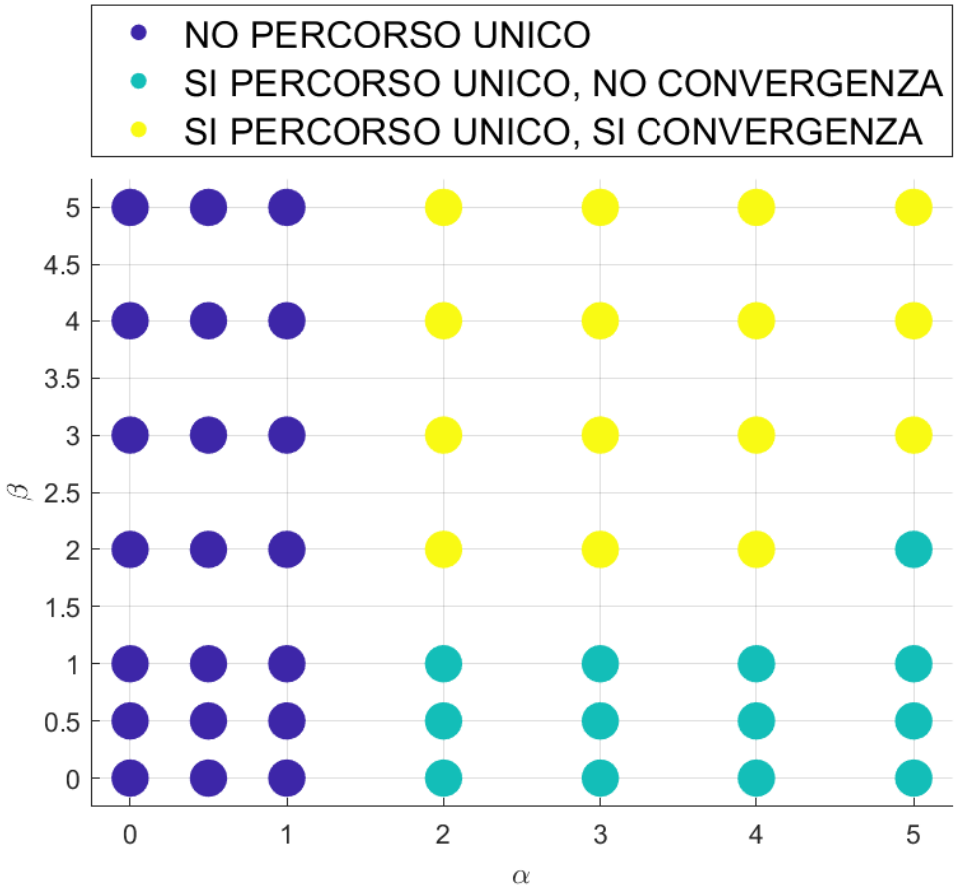


Figura 3.4: Percorso unico e convergenza al minimo. I risultati sono mediati su 20 trials: si parla di percorso unico e di convergenza per le configurazioni per cui sono stati osservati almeno per il 75% dei trials. Tutte e sole le configurazioni con $\alpha \geq 2$ sono quelle per cui si ha percorso unico; tra di esse, generalmente, per $\beta \geq 2$ si ha anche convergenza.

I risultati delle simulazioni per il percorso unico si possono interpretare come segue: per α sufficientemente elevato, come detto, l'intensità di ferormone diventa preponderante nella scelta del nodo di destinazione, il che induce tutti gli agenti a percorrere sempre gli stessi cicli; se, inoltre, β è elevato, i cicli su cui la ricerca si stabilizza sono, come già evidenziato in Fig.(3.3), quelli minimi. Sembra dunque che esista un valore α_0 t.c. si osserva percorso unico $\forall\ \alpha > \alpha_0$, $\forall\ \beta \in [0,5]$ (che esso sia il minimo o meno). Dalla Fig.(3.4), è inoltre evidente che $\alpha_0 \in (1,2)$. E' stata svolta quindi un'analisi con $\alpha \in [1,2]$ e β fissato a 1 con lo scopo di individuare α_0: nei risultati, illustrati in Fig.(3.5), è riportato anche il numero di iterazioni dopo le quali, eventualmente, gli agenti iniziano a percorrere tutti cicli simili.

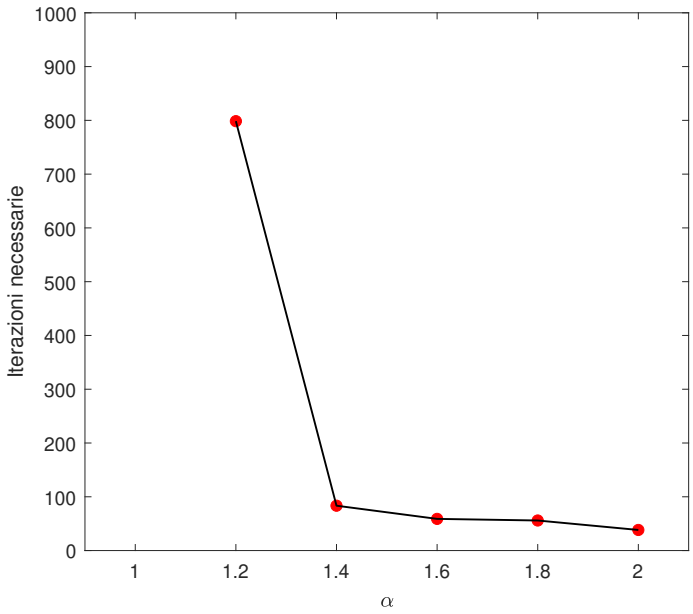


Figura 3.5: Numero di iterazioni necessarie per osservare eventualmente comportamento di percorso unico, per $\alpha \in [1,2]$ e $\beta = 1$, con $NI_{max} = 5000$. Solo con $\alpha = 1$ non si ha, come già evidenziato in Fig.(3.4), percorso unico, dunque non è presente alcun valore di iterazioni necessarie. Si nota che, all'aumentare di α, è sufficiente un numero sempre decrescente di iterazioni, e che la variazione è particolarmente significativa nella metà sinistra dell'intervallo.

Osservando la Fig.(3.5), sembra che si abbia percorso unico $\forall\ \alpha > \alpha_0 = 1$ (ma non per 1), a patto di impostare un valore sufficientemente elevato di iterazioni totali NI_{max}. Per tentare di confermare questa ipotesi da un punto di vista analitico, si può partire dalle Eq.(2.3) e (3.1) con il fine di calcolare qualcosa in più sulle dinamiche collettive degli agenti al variare di α. L'obiettivo è dimostrare che il comportamento collettivo del gruppo di agenti cambi non appena α supera il valore $\alpha_0 = 1$. Innanzitutto, si ricordi che, nell'inizializzazione dell'algoritmo, l'intensità $\tau_{i,j}(0)$ di ferormone sui segmenti è pari a 10 su ogni segmento $s_{i,j}$ di $\mathcal{G}_4$. Tuttavia, questa omogeneità iniziale va via via a decadere man mano che gli agenti si muovono secondo (2.2): su alcuni segmenti inizia infatti ad accumularsi un'intensità maggiore, mentre altri vengono percorsi meno frequentemente e, di conseguenza, l'intensità su di essi diminuisce a causa dell'evaporazione. In particolare, sicuramente ci sarà un certo istante t_0 e due segmenti di $\mathcal{G}_4$, che saranno denotati con $s_{i,j}$ ed $s_{i,k}$, t.c.

$$\tau_{i,j}(t_0) > \tau_{i,k}(t_0). \tag{3.2}$$

A questo proposito, in Eq.(3.1) si è già definito il rapporto di preferibilità $r_i^{j,k}(t)$ tra due nodi c_j e c_k a partire da un terzo nodo c_i al tempo t: l'analisi che segue è volta a studiare la variazione nel lungo termine del generico $r_i^{j,k}(t)$, partendo dalla condizione (3.2), in

funzione del valore di α. Per comodità di calcolo, si è posto $t_0 = 0$, ovvero si considera l'evoluzione della ricerca a partire da quell'istante. Nelle probabilità $p_{i,j}^m$ di transizione, inoltre, si ometterà l'apice m per semplicità di notazione, dal momento che i calcoli che seguono valgono per tutti gli agenti $\{a_m \mid m \in \{1, ..., M\}\}$ che non hanno visitato il nodo di destinazione.

Sia quindi

$$\begin{cases} \tau_{i,j}(0) = F\tau_{i,k}(0), & F > 1 \\ \eta_{i,j} = R\,\eta_{i,k}, & R > 0. \end{cases} \tag{3.3}$$

Combinando (3.1) e (3.3), si ha:

$$r_i^{j,k}(0) = \frac{p_{i,j}(0)}{p_{i,k}(0)} = F^\alpha R^\beta; \tag{3.4}$$

pertanto, la frequenza attesa di percorrimento del segmento $s_{i,j}$ nell'iterazione successiva è $F^\alpha R^\beta$ volte più alta di quella del segmento $s_{i,k}$. Proprio tali frequenze attese di percorrimento sono strettamente associate alle intensità di ferormone che verranno rilasciate sui due segmenti, indicate rispettivamente con $\mathcal{I}(\tau_{i,j})$ e $\mathcal{I}(\tau_{i,k})$, che si possono calcolare da (2.4). Per ciascuna di esse, si tratta in realtà di una somma di componenti in generale diverse, esplicitate in (2.5): in particolare, il contributo di ogni agente a_m dipende dalla lunghezza $L_{\mathbf{c}_m}$ del ciclo che egli sta percorrendo; tuttavia, si è utilizzata un'approssimazione con un ciclo fissato $\tilde{\mathbf{c}}$, in modo da rendere comparabili $\mathcal{I}(\tau_{i,j})$ e $\mathcal{I}(\tau_{i,k})$. In effetti, in $\mathcal{G}_4$, tutti i possibili cicli hanno lunghezze dello stesso ordine di grandezza. Da queste considerazioni si ottiene:

$$\frac{\mathcal{I}(\tau_{i,j})}{\mathcal{I}(\tau_{i,k})} \approx F^\alpha R^\beta. \tag{3.5}$$

Da (2.3) e (3.5) si ha perciò:

$$\frac{\tau_{i,j}(n)}{\tau_{i,k}(n)} = \frac{\rho\tau_{i,j}(0) + \mathcal{I}(\tau_{i,j})}{\rho\tau_{i,k}(0) + \mathcal{I}(\tau_{i,k})} \approx \frac{\rho\tau_{i,j}(0) + F^\alpha R^\beta \mathcal{I}(\tau_{i,k})}{k_1 \mathcal{I}(\tau_{i,k})} \approx \frac{F^\alpha R^\beta}{k_1}, \tag{3.6}$$

dove al numeratore si ipotizza che l'intensità iniziale $\tau_{i,j}(0)$ sia trascurabile rispetto al termine in F^α e al denominatore si ingloba $\rho\tau_{i,k}(0)$ in $\mathcal{I}(\tau_{i,k})$ con la costante moltiplicativa $k_1 > 0$. Da (3.1) e (3.6) si ottiene poi:

$$r_i^{j,k}(n) = \frac{p_{i,j}(n)}{p_{i,k}(n)} \approx \left[\frac{F^\alpha R^\beta}{k_1}\right]^\alpha R^\beta = \frac{F^{(\alpha^2)}}{k_1^\alpha} R^{(\alpha\beta+\beta)}. \tag{3.7}$$

Tramite considerazioni analoghe a quelle fatte per (3.5), si deduce:

$$\frac{\mathcal{I}(\tau_{i,j})}{\mathcal{I}(\tau_{i,k})} \approx \frac{F^{(\alpha^2)}}{k_1^\alpha} R^{(\alpha\beta+\beta)}, \tag{3.8}$$

da cui (nuovamente per (2.3)):

$$\begin{aligned}\frac{\tau_{i,j}(2n)}{\tau_{i,k}(2n)} = \\ = \frac{\rho\tau_{i,j}(n)+\mathcal{I}(\tau_{i,j})}{\rho\tau_{i,k}(n)+\mathcal{I}(\tau_{i,k})} \approx \frac{\rho\tau_{i,j}(n)+\left[\frac{F^{(\alpha^2)}}{k_1^\alpha}R^{(\alpha\beta+\beta)}\right]\mathcal{I}(\tau_{i,k})}{k_2\mathcal{I}(\tau_{i,k})} \approx \frac{F^{(\alpha^2)}R^{(\alpha\beta+\beta)}}{k_1^\alpha k_2},\end{aligned} \tag{3.9}$$

dove, analogamente a (3.6), il termine di intensità $\tau_{i,j}(n)$ è stato considerato trascurabile rispetto a $F^{(\alpha^2)}$ e $\rho\tau_{i,k}(n)$ è stato inglobato in $\mathcal{I}(\tau_{i,k})$ tramite la costante moltiplicativa $k_2 > 0$. Analogamente a (3.7):

$$r_i^{j,k}(2n) = \frac{p_{i,j}(2n)}{p_{i,k}(2n)} \approx \left[\frac{F^{(\alpha^2)}R^{(\alpha\beta+\beta)}}{k_1^\alpha k_2}\right]^\alpha R^\beta = \frac{F^{(\alpha^3)}}{k_1^{(\alpha^2)}k_2^\alpha}R^{(\alpha^2\beta+\alpha\beta+\beta)}. \tag{3.10}$$

Generalizzando l'Eq.(3.10) per $t \in \mathbb{N}$, si ottiene:

$$r_i^{j,k}(tn) = \frac{p_{i,j}(tn)}{p_{i,k}(tn)} \approx \frac{F^{(\alpha^{t+1})}}{\prod_{i=0}^{t-1} k_{i+1}^{(\alpha^{t-i})}} R^{\left(\beta\sum_{i=0}^{t}\alpha^i\right)}, \tag{3.11}$$

il cui termine dominante è $F^{(\alpha^{t+1})}$. Il limite della successione in (3.11) è il rapporto di preferibilità a lungo termine tra i nodi c_j e c_k a partire dal nodo c_i, dunque esprime la frequenza relativa con la quale i segmenti $s_{i,j}$ e $s_{i,k}$ vengono percorsi nel lungo periodo, a partire dal rapporto $F := \tau_{i,j}(0)/\tau_{i,k}(0)$ tra le intensità di ferormone inizialmente presenti e dal rapporto $R := \eta_{i,j}/\eta_{i,k}$ tra le visibilità. Si ha:

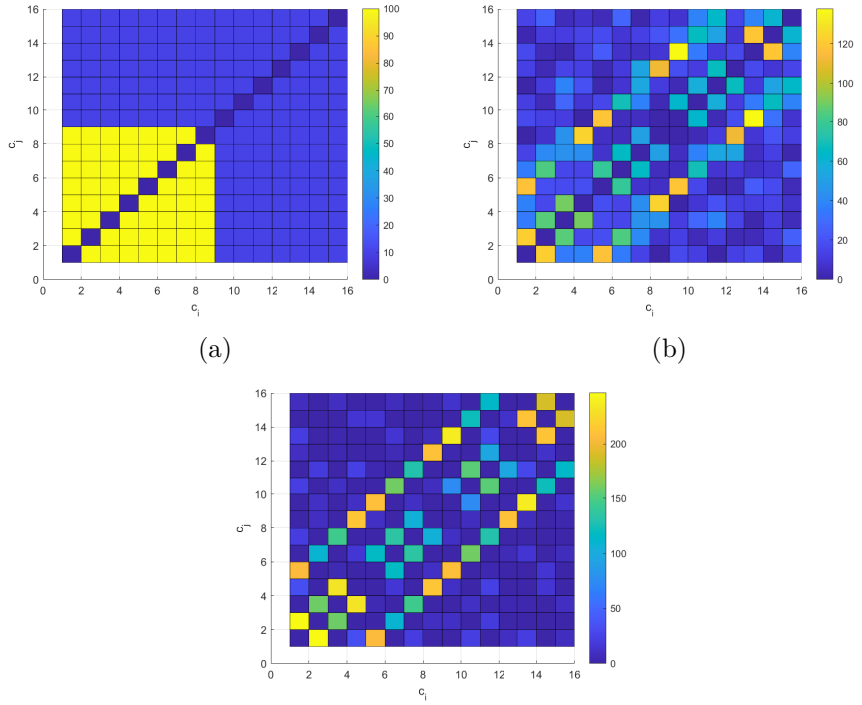
$$\lim_{t\to+\infty} r_i^{j,k}(tn) \begin{cases} = +\infty & \alpha > 1 \\ < +\infty & \alpha < 1. \end{cases} \tag{3.12}$$

Per $\alpha = 1$, invece, (3.11) diventa

$$r_i^{j,k}(tn) \approx \frac{FR^{\beta(t+1)}}{\prod_{i=0}^{t-1} k_{i+1}} : \tag{3.13}$$

la successione sembra dunque divergere, contrariamente al risultato sperimentale, sebbene in modo più lento che per $\alpha > 1$. Tuttavia, tale deduzione non può essere una certezza, a causa del fatto che non si conosce la natura della successione $(k_n)_{n\in\mathbb{N}}$: potrebbe essere tale da "compensare" l'andamento esponenziale di (3.13) ma non quello super-esponenziale di (3.11). Se così fosse, questo spiegherebbe perché è sufficiente un valore di α di poco superiore a 1 per avere percorso unico: se il limite in (3.12) è $+\infty$, i segmenti che nelle prime fasi sono preferibili a causa di intensità di ferormone maggiori lo diventano sempre di più; questa dinamica non si verifica invece quando il limite è finito. Si noti inoltre che, in ogni caso, per $\alpha \neq 1$ il verificarsi o meno del percorso unico non dipende dal valore di β.

Per quanto riguarda $\alpha < 1$, sia l'Eq.(3.12) che già la Fig.(3.4) evidenziano che il modello non presenta percorso unico. Tuttavia, ci si può domandare come funzioni la dinamica di percorrimento dei segmenti nel lungo termine partendo da una distribuzione molto disomogenea di ferormone: in altre parole, ci si chiede se il ferormone, pur non generando percorso unico, mantenga comunque una certa disomogeneità o diventi via via più omogeneo sui segmenti di $\mathcal{G}_4$. Si noti che l'Eq.(3.12) non fornisce una risposta in merito, poiché non si conosce il valore esatto del limite per $\alpha < 1$. I risultati empirici sono invece presentati in Fig.(3.6).



(a) (b)

(c)

Figura 3.6: Distribuzione dell'intensità di ferormone sui segmenti che connettono il nodo c_i (in ascissa) e c_j (in ordinata). (a) Distribuzione iniziale. (b) Distribuzione dopo 5000 iterazioni, con $\alpha = 0.8$ e $\beta = 1$: la disomogeneità iniziale si è completamente annullata. (c) Distribuzione dopo 5000 iterazioni, con $\alpha = 0.8$ e $\beta = 5$: anche in questo caso il ferormone è più omogeneo che per $t = 0$, ma meno che in (b).

Dalla Fig.(3.6) emerge che è sufficiente un valore di α non molto < 1 per rendere la distribuzione di ferormone più omogenea. Il ruolo di β è secondario in questo senso e mantiene una parziale disomogeneità, favorendo i percorsi più brevi per le ragioni già discusse.

3.3 Dipendenza dal numero di agenti

In tutte le simulazioni svolte finora, il numero M di agenti è stato sempre impostato uguale al numero n di nodi del grafo, dunque 16 per $\mathcal{G}_4$. In questa sezione, è stata invece eseguita un'analisi per studiare i risultati di convergenza dell'algoritmo al variare di M. I risultati sono illustrati in Fig.(3.7.)

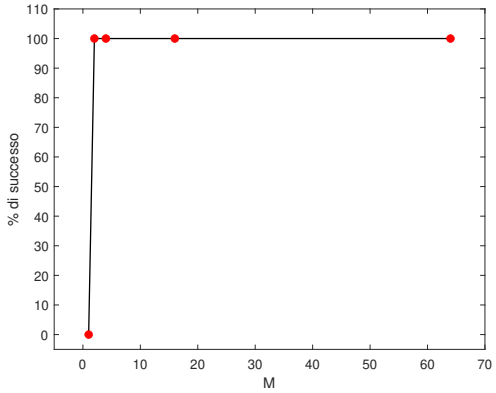


Figura 3.7: Percentuale di successo dell'algoritmo nell'individuare il ciclo minimo reale $\mathbf{c}^*$ al variare di M entro 5000 iterazioni. I parametri sono impostati ai valori standard, dunque $\{\alpha = 1, \beta = 1, \rho = 0.7\}$. La ricerca è totalmente efficace per ogni valore di M considerato, salvo che per $M = 1$, situazione in cui il ciclo minimo non viene mai rilevato.

Emerge che i risultati di convergenza cambiano radicalmente passando anche solo da $M = 1$ a $M = 2$. Per cercare di comprenderne la ragione, si è analizzato il comportamento di percorso unico nei due casi: in Fig.(3.8) sono illustrati i risultati.

Dalla Fig.(3.8) si osserva che l'algoritmo presenta comportamento di percorso unico sia per $M = 1$ che per $M = 2$. Nel primo caso, si tratta evidentemente di un percorso non minimo, dal momento che l'algoritmo non converge. Anche nel secondo caso, in realtà, il ciclo unico trovato è molto raramente quello minimo, nonostante si fosse osservata in Fig.(3.7) la convergenza dell'algoritmo per $M = 2$: evidentemente, gli agenti individuano il ciclo minimo, ma successivamente convergono a un altro percorso unico. Questo fenomeno si potrebbe interpretare come segue: per $M = 1$, gli agenti si stabiliscono rapidamente su un percorso unico non minimo e non riescono quindi poi a rilevare il ciclo minimo; per $M = 2$, invece, il ciclo unico emerge molto più tardi, quindi gli agenti nel frattempo riescono a rilevare il percorso minimo.

La dinamica del ferormone in Eq.(2.3) potrebbe spiegare il motivo per cui il percorso unico emerge prima in un caso che nell'altro. Si consideri innanzitutto $M = 1$. Sia $\tau_0 := \tau_{i,j}(0)\ \forall\ i, j$ l'intensità iniziale di ferormone, uguale su tutti i segmenti; sia inoltre $\Delta\tau_k$ l'intensità rilasciata, all'iterazione k, su ogni segmento percorso da a_1: viene sostituita

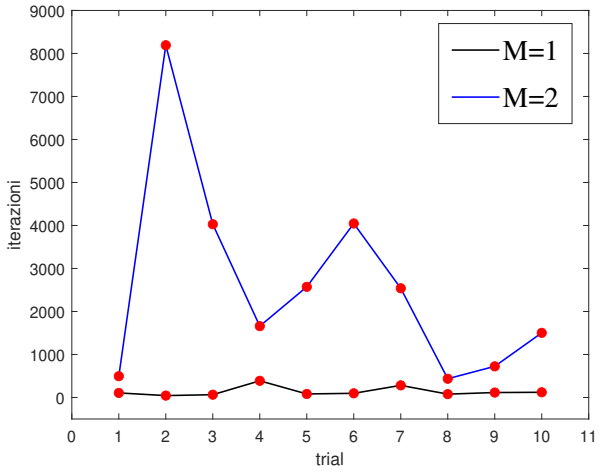


Figura 3.8: Iterazioni necessarie per osservare percorso unico per $M \in \{1,2\}$ in 10 trials. Va precisato che non si è arrestato l'algoritmo al rilevamento dell'ottimo reale, in modo da verificare se gli agenti si stabiliscano su un percorso unico a prescindere dalla convergenza. Si ha percorso unico per entrambi i valori di M, ma gli agenti vi arrivano molto prima per $M = 1$.

la notazione $\mathcal{I}(\tau_{i,j})$ di (2.3), poiché, essendoci un solo agente, l'intensità rilasciata è la stessa su ogni segmento percorso a parità di iterazione. Si ha allora:

$$\tau_{i,j}(n) = \begin{cases} \rho\tau_0 & \text{se } s_{i,j} \text{ non è stato percorso} \\ \rho\tau_0 + \Delta\tau_1 & \text{se } s_{i,j} \text{ è stato percorso;} \end{cases} \tag{3.14}$$

e quindi:

$$\tau_{i,j}(2n) = \begin{cases} \rho^2\tau_0 & \text{se } s_{i,j} \text{ non è stato mai percorso} \\ \rho^2\tau_0 + \rho\Delta\tau_1 & \text{se } s_{i,j} \text{ è stato percorso per } k = 1 \\ \rho^2\tau_0 + \Delta\tau_2 & \text{se } s_{i,j} \text{ è stato percorso per } k = 2 \\ \rho^2\tau_0 + \rho\Delta\tau_1 + \Delta\tau_2 & \text{se } s_{i,j} \text{ è stato percorso per } k = 1,2. \end{cases} \tag{3.15}$$

Si noti che $\Delta\tau_1$ è, in generale, diverso da $\Delta\tau_2$, poiché l'intensità rilasciata su un ciclo dipende, come evidenziato in (2.5), dalla lunghezza del ciclo stesso. Generalizzando, l'intensità di ferormone su un segmento $s_{i,j}$ per $t \in \mathbb{N}$ si può scrivere come:

$$\tau_{i,j}(tn) = \rho^t\tau_0 + \sum_{k=1}^{t} q_k^{i,j}\ \rho^{(t-k)}\ \Delta\tau_k, \tag{3.16}$$

dove

$$q_k^{i,j} = \begin{cases} 1 & \text{se } s_{i,j} \text{ è stato percorso all'iterazione } k \\ 0 & \text{altrimenti,} \end{cases} \tag{3.17}$$

$\forall\, k \in \{1, ..., t\}$. Dall'Eq.(3.16) si deduce che, dopo t iterazioni, ci sono potenzialmente 2^t intensità diverse di ferormone sui segmenti: una per ogni possibile valore della successione $(q_k^{i,j})_{k\in\{1,...,t\}}$. Sia allora $\mathcal{K}(t, M)$ la cardinalità dell'insieme delle possibili intensità di ferormone su $\mathcal{G}_4$ all'iterazione k, quando vi sono M agenti: si può dunque scrivere

$$\mathcal{K}(t,1) = 2^t. \tag{3.18}$$

Si consideri ora la dinamica del ferormone per $M = 2$: sia $\Delta\tau_{k,m}$ l'intensità di ferormone rilasciata all'iterazione k dall'agente a_m, per $m \in \{1,2\}$. Si ha:

$$\tau_{i,j}(n) = \begin{cases} \rho\tau_0 & \text{se } s_{i,j} \text{ non è stato percorso} \\ \rho\tau_0 + \Delta\tau_{1,1} & \text{se } s_{i,j} \text{ è stato percorso da } a_1 \\ \rho\tau_0 + \Delta\tau_{1,2} & \text{se } s_{i,j} \text{ è stato percorso da } a_2 \\ \rho\tau_0 + \Delta\tau_{1,1} + \Delta\tau_{1,2} & \text{se } s_{i,j} \text{ è stato percorso da } a_1, a_2. \end{cases} \tag{3.19}$$

Per $\tau_{i,j}(2n)$ vi saranno 4 possibilità per ogni opzione di (3.19), a seconda che il segmento $s_{i,j}$ sia percorso da nessuno, da entrambi o da uno dei due agenti all'iterazione $k = 2$, per un totale di 16 combinazioni. Generalizzando per $t \in \mathbb{N}$:

$$\tau_{i,j}(tn) = \rho^t\tau_0 + \sum_{k=1}^{t}\sum_{m=1}^{2} q_{k,m}^{i,j}\, \rho^{(t-k)}\, \Delta\tau_{k,m}, \tag{3.20}$$

dove

$$q_{k,m}^{i,j} = \begin{cases} 1 & \text{se } s_{i,j} \text{ è stato percorso in } k \text{ da } a_m \\ 0 & \text{altrimenti,} \end{cases} \tag{3.21}$$

$\forall\, \{k, m\} \in [\{1, ..., t\} \times \{1,2\}]$. Da (3.20) si deduce:

$$\mathcal{K}(t,2) = 2^{2t}. \tag{3.22}$$

Generalizzando l'analisi per $M > 2$, l'Eq.(3.20) diventa:

$$\tau_{i,j}(tn) = \rho^t\tau_0 + \sum_{k=1}^{t}\sum_{m=1}^{M} q_{k,m}^{i,j}\, \rho^{(t-k)}\, \Delta\tau_{k,m}, \tag{3.23}$$

dove $q_{k,m}^{i,j}$ è analogo a (3.21), ma con $m \in \{1, ..., M\}$; si ha perciò:

$$\mathcal{K}(t, M) = 2^{Mt} = [\mathcal{K}(t,1)]^M. \tag{3.24}$$

Si può dunque concludere che il numero di valori diversi assumibili dall'intensità di ferormone sui segmenti di $\mathcal{G}_4$, a parità di iterazione, cresce esponenzialmente con il numero M di agenti. Per questa ragione, si può pensare che, a parità di iterazione, la scelta della destinazione successiva sia più "deterministica" per $M = 1$: i possibili valori di intensità di ferormone sono "pochi" ed è quindi frequente che un agente debba scegliere tra due segmenti con intensità molto diverse. Questo comportamento rafforza i segmenti già molto percorsi e fa stabilire più rapidamente gli agenti su un percorso unico.

Capitolo 4

Risultati per la topologia Oliver30

La griglia 4×4, come detto, è una topologia regolare, della quale si conoscono i cicli minimi per via analitica. In questo capitolo viene introdotta invece una topologia più irregolare, con $n = 30$ punti, nota in letteratura con il nome di Oliver30 (Holland [1975]) e illustrata in Fig.(4.1). I cicli minimi di Oliver30 non sono noti per via analitica; il risultato migliore

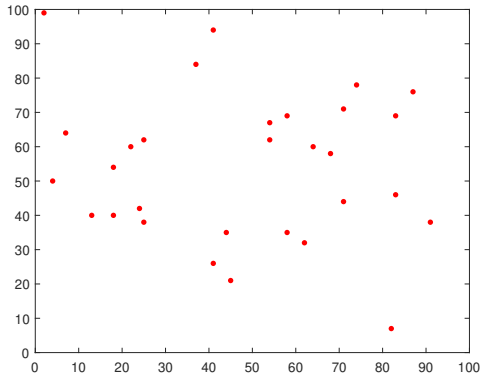


Figura 4.1: Oliver30

presente in letteratura è costituito dai cicli simili a quello in Fig.(4.2), da qui in avanti "cicli minimi noti". Si è esaminato se esista qualche combinazione di valori dei parametri $\{\alpha, \beta, \rho\}$ per la quale il gruppo di agenti riesca a individuare uno dei cicli minimi noti. Analogamente al caso della griglia, l'analisi è stata eseguita variando di volta in volta un solo parametro e lasciando gli altri ai valori standard predefiniti: $\{\alpha = 1, \beta = 1, \rho = 0.7\}$. I risultati in Fig.(4.3) mostrano la dipendenza del minimo rilevato dai parametri del modello. Gli andamenti osservati per α e β possono essere interpretati con considerazioni analoghe a quelle espresse per la topologia di griglia: un valore troppo basso di α annulla l'utilità del ferormone, mentre uno troppo alto forza la ricerca sui segmenti più percorsi

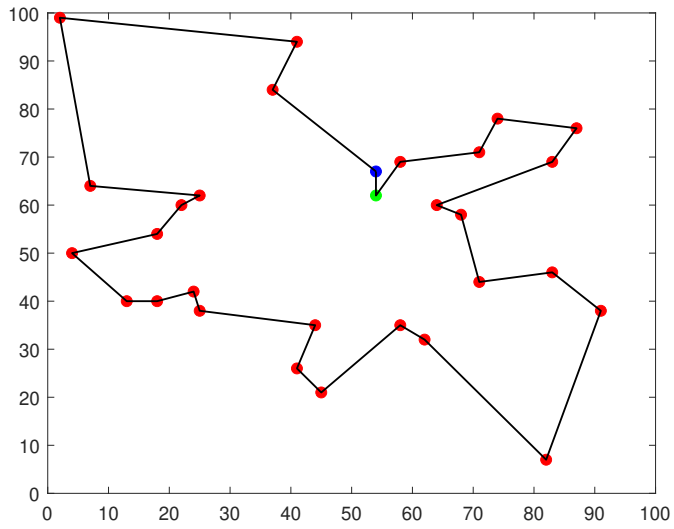


Figura 4.2: Uno dei cicli minimi noti per il problema Oliver30, di lunghezza pari a 423.74: nella convenzione adottata qui e successivamente, il nodo di partenza è rappresentato in blu e il secondo (mostrato per indicare il verso di percorrenza) è rappresentato in verde.

nelle prime iterazioni; un valore troppo basso di β toglie un riferimento utile nelle prime fasi (in cui il ferormone è poco), mentre uno troppo elevato impedisce il passaggio sui segmenti più lunghi. A quest'ultimo proposito, si ricordi che, per $\mathcal{G}_4$, esiste $\beta_0(=3)$ t.c. la ricerca converge $\forall\ \beta \geq \beta_0$. Nel problema Oliver30, invece, l'effetto descritto per β elevato ha la conseguenza opposta: in questo caso, infatti, non sempre i segmenti più brevi appartengono ai cicli minimi noti, come evidente dalla Fig.(4.2), il che rende la scelta di β elevato poco efficace. Per quanto riguarda il valore di ρ:

- se è molto vicino a 1, è forte la tendenza degli agenti a concentrarsi sui segmenti già percorsi, perché la traccia di ferormone resta intensa: l'effetto è simile a quello che si ha per α elevato. Lo svantaggio, analogamente, è l'inefficacia dell'algoritmo a sondare adeguatamente i numerosi percorsi possibili;

- se, al contrario, ρ è molto prossimo a 0, la strategia del ferormone viene in buona misura invalidata dal fenomeno di evaporazione, che impedisce di tenere traccia dei cammini migliori.

Si noti, tuttavia, che l'effetto di $\rho = 0$ non è esattamente lo stesso di $\alpha = 0$. L'aggiornamento di $\tau_{i,j}(t)$ è infatti modellizzato come il processo discreto in (2.3), verificandosi al termine di ogni iterazione: di conseguenza, quando il generico agente a_m compie la scelta del nodo di destinazione alla generica iterazione k, è ancora completamente presente il ferormone rilasciato sui segmenti all'iterazione $k-1$ (mentre non c'è traccia diretta delle scelte effettuate alle iterazioni $\{1, ..., k-2\}$).

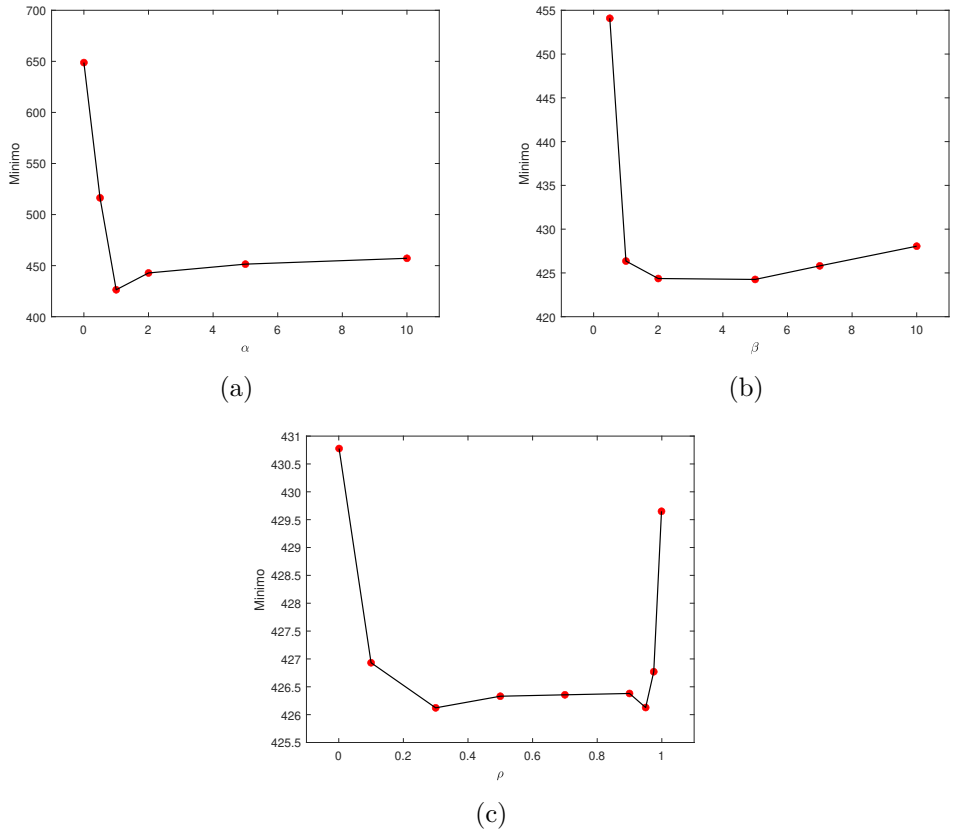


Figura 4.3: Minimo individuato dall'algoritmo al variare di α (a), β (b) e ρ (c). I valori rappresentati sono mediati su 20 trials e il numero massimo NI_{max} di iterazioni è stato impostato a 5000. Gli agenti non riescono a individuare i cicli minimi noti per nessuna delle combinazioni di parametri simulate. La scelta di ρ influisce sul risultato in misura minore rispetto ad α e a β. La ricerca è più efficace per valori intermedi: più precisamente, gli intervalli ottimali di $\{\alpha, \beta, \rho\}$ sono rispettivamente {(0.5,2),(1,7),(0.1,0.975)}

Capitolo 5

Introduzione di agenti elitari

Nel capitolo 4 si è osservato che l'algoritmo non converge ai cicli minimi noti di Oliver30 per nessuna combinazione dei parametri $\{\alpha, \beta, \rho\}$: una strategia che si può implementare per tentare di indurre la convergenza consiste nell'introduzione dei cosiddetti "agenti elitari". L'idea alla base degli agenti elitari è quella di rinforzare in modo intelligente la dinamica del ferormone, procedendo come segue (si ricordi che, al termine di ogni iterazione $k \in \{1, ..., NI_{max}\}$, la variabile $\mathbf{c}^*$ contiene il ciclo più breve individuato dagli M agenti durante le iterazioni $\{1, ..., k\}$):

- nella fase di inizializzazione dell'algoritmo, viene stabilito un numero n_e di agenti elitari;
- al termine di ogni iterazione $k \in \{1, ..., NI_{max}\}$, gli n_e agenti elitari percorrono $\mathbf{c}^*$.

L'effetto della strategia descritta è quello di spargere ulteriore ferormone sui segmenti che costituiscono il ciclo migliore rilevato fino a quel momento. Questo processo può apparire simile all'aumentare il parametro α, tuttavia si tratta di un rafforzamento più mirato, che incentiva il passaggio esclusivamente su un ciclo alla volta, quello che per il momento appare preferibile. L'efficacia della tecnica degli agenti elitari è stata valutata studiando la convergenza o meno dell'algoritmo ai cicli minimi noti di Oliver30 al variare di n_e: il caso $n_e = 0$ corrisponde evidentemente all'algoritmo standard. I valori scelti per i parametri sono quelli che apparivano ottimali in Fig.(4.3): ($\{\alpha = 1, \beta = 5, \rho = 0.95\}$). Oltre alla capacità o meno degli agenti di individuare un ciclo minimo noto, si è calcolato il numero di iterazioni necessarie a rilevarlo. I risultati sono illustrati in Fig.(5.1). Gli agenti elitari rendono l'algoritmo da totalmente inefficace a totalmente efficace: questo processo avviene in modo graduale, attraverso l'aggiunta dei primi quattro agenti. L'introduzione di agenti elitari dopo il quarto non influenza la capacità di rilevamento dei cicli minimi, ma permette di accelerare la convergenza fino a un punto di minimo, dopo il quale essa rallenta nuovamente.

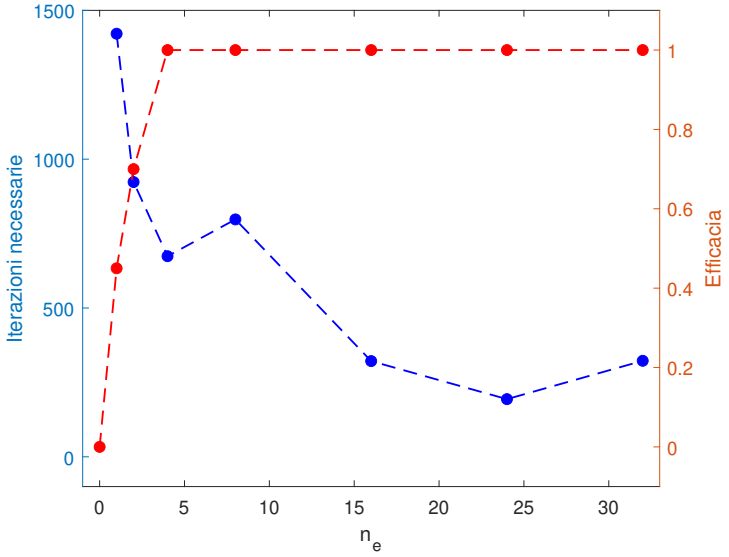


Figura 5.1: Efficacia dell'algoritmo nel rilevare uno dei cicli minimi noti di Oliver30 (in rosso) e numero medio di iterazioni necessarie per farlo (in blu) al variare della quantità n_e di agenti elitari. L'efficacia è intesa come frazione dei trials in cui la ricerca ha avuto successo (20 trials totali per ogni valore di n_e). Si evidenzia che la strategia è perfettamente efficace per $n_e \geq 4$, mentre nel caso standard non vengono mai rilevati cicli minimi noti. La velocità di convergenza sembra massima per $n_e \in (16{,}32)$.

Capitolo 6

Nuove varianti

L'introduzione degli agenti elitari presentata nel capitolo 5 consiste in una variante dell'algoritmo standard, nota in letteratura, che ha il fine di migliorare i risultati di convergenza. In questo capitolo verranno presentate nuove varianti, elaborate autonomamente con lo stesso scopo e applicate al problema Oliver30.

6.1 Variante con inerzia

Si può supporre che alcuni agenti abbiano la tendenza, una volta percorso un segmento, a proseguire per un tratto nella stessa direzione e nello stesso verso, anziché selezionare la destinazione successiva in base all'Eq.(2.2). Questo fenomeno può essere letto come un comportamento di inerzia locale, che permetterebbe agli agenti di continuare a muoversi nella direzione che stanno ritenendo ottima. In realtà, nella definizione del problema, ogni movimento compiuto dagli agenti inizia e finisce comunque in un nodo del grafo; per tale ragione, il termine di inerzia locale è stato modellizzato come segue: la legge di transizione (2.2) è sostituita da

$$p_{i,j}^{m}(t) = \begin{cases} (2.2) & \text{se il prossimo passo di } a_m \text{ non è secondo inerzia} \\ \delta_{j,z} & \text{se il prossimo passo di } a_m \text{ è secondo inerzia} \end{cases} \tag{6.1}$$

con

$$z := \underset{h \in \mathcal{G}^m(t)}{\operatorname{argmin}}\, d(h, p^*), \tag{6.2}$$

dove p^* è la corrente destinazione ideale di a_m, che egli raggiungerebbe dal nodo corrente c_i percorrendo un vettore uguale in modulo, direzione e verso al suo ultimo passo. L'agente a_m si muove secondo inerzia, allo step $s \in \{1, ..., n\}$ dell'iterazione $k \in \{1, ..., NI_{max}\}$, se

$$v_{s,k}^{m} < f, \tag{6.3}$$

dove $v_{s,k}^{m} \in [0,1]$ è estratto da una variabile uniforme $\mathcal{U}([0,1])$, mentre $f \in [0,1]$ è un valore settato in fase di inizializzazione che esprime il valore atteso della frazione di movimenti totali che avvengono su base inerziale. Lo scopo della dinamica introdotta è dunque

imporre che una frazione vicina a f dei movimenti complessivi degli agenti sia dettata da inerzia, compatibilmente con i vincoli del problema. In Fig.(6.1) è illustrato un ciclo ottenuto dall'algoritmo con una forte inerzia locale.

Se si osservano pattern simili a quello in Fig.(6.1), il metodo dell'inerzia rischia di divenire controproducente: il vincolo di percorrimento di tutti gli n nodi può invalidare la strategia. La Fig.(6.2) mostra il minimo individuato dall'algoritmo al variare di f. I parametri sono stati settati ai valori standard.

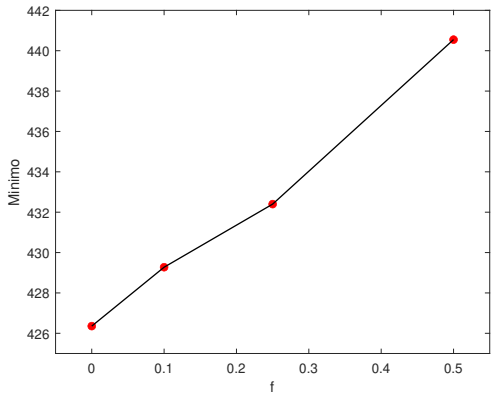


Figura 6.2: Minimo rilevato dagli agenti al variare di f. Più forte è il termine di inerzia, peggiore è la ricerca.

6.2 Variante con movimenti casuali

Un'altra strategia per provare a migliorare la convergenza dell'algoritmo consiste nell'introduzione di movimenti casuali. Come nel caso dell'inerzia, occorre tuttavia modellizzare tali movimenti in modo che abbiano sempre un nodo del grafo come partenza e uno come destinazione. L'Eq.(2.2) è sostituita da

$$p_{i,j}^{m}(t) = \begin{cases} (2.2) & \text{se il prossimo passo di } a_m \text{ non è casuale} \\ \delta_{j,z} & \text{se il prossimo passo di } a_m \text{ è casuale} \end{cases} \tag{6.4}$$

con

$$z := \underset{h \in \mathcal{G}^m(t)}{\operatorname{argmin}}\, d(h, p^*), \tag{6.5}$$

dove p^* è la corrente destinazione ideale di a_m, per cui vale

$$\begin{cases} x_{p^*} = x_i + l\ cos(\theta) \\ y_{p^*} = y_i + l\ sin(\theta), \end{cases} \tag{6.6}$$

con $l \in [0, l_{max}]$ estratto da una variabile uniforme $\mathcal{U}([0, l_{max}])$ e $\theta \in [0,2\pi]$ estratto da una variabile uniforme $\mathcal{U}([0,2\pi])$. A sua volta, l_{max} è la metà della distanza massima tra

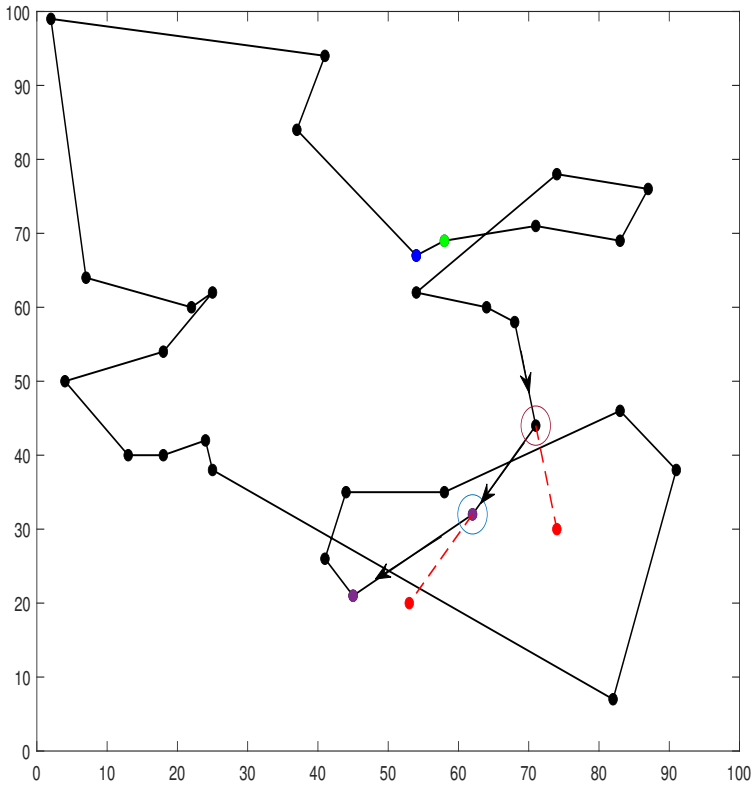


Figura 6.1: Uno dei cicli individuati dall'algoritmo con inerzia locale, per $f = 0.95$. I nodi rappresentati in blu e in verde, rispettivamente il primo e il secondo del ciclo, indicano il verso di percorrenza. Si considerino i due nodi cerchiati: quando un agente si trova su uno dei due nodi e sta percorrendo il ciclo in figura, se si comporta secondo inerzia ha come destinazione ideale (p^* in formula (6.2)) uno dei due nodi rossi. Le destinazioni reali (z in formula (6.2)) sono quindi rappresentate dai nodi viola e il ciclo prosegue perciò nella direzione indicata dalle frecce nere. Quando si esauriscono i nodi disponibili in direzioni vicine a quelle delle frecce, gli agenti non possono che "tornare indietro" per completare il ciclo. Il risultato è un ciclo molto più lungo di quelli minimi noti.

due nodi del grafo. La selezione degli agenti che si muovono in modo casuale avviene secondo un meccanismo identico a quello dei movimenti inerziali descritto nella sezione precedente. La dinamica introdotta ha l'effetto di imporre che una frazione vicina a un valore predefinito f dei movimenti totali degli agenti sia casuale, sia come direzione dello spazio che come lunghezza. In Fig.(6.3) è illustrato l'andamento della ricerca al variare di f; il caso $f = 0$ corrisponde evidentemente all'algoritmo standard. La strategia si è rivelata inefficiente.

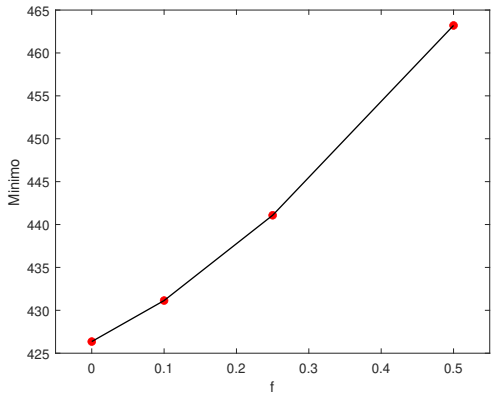


Figura 6.3: Minimo individuato dagli agenti al variare della frazione f di movimenti casuali attesi. Come nel caso dell'inerzia locale, la ricerca è tanto peggiore quanto più è forte il termine introdotto.

6.3 Variante con rimozione di nodi dal modello

Una strategia che si può implementare con lo scopo di permettere agli agenti di individuare nuovi cicli consiste nella rimozione temporanea di uno o più nodi dal grafo $\mathcal{G}$. In questo modo, si impedisce il passaggio degli agenti su tutti i segmenti di $\mathcal{G}$ che hanno il/i nodo/i rimosso/i come estremo/i, imponendo quindi il percorrimento di altri segmenti. La tecnica descritta viene implementata con i seguenti passi quando riguarda un solo nodo, che sarà indicato con c_r:

1. rimozione temporanea del nodo c_r dal grafo $\mathcal{G}$, che trasforma $\mathcal{G}$ in un nuovo grafo $\mathcal{G}'$;

2. esecuzione dell'algoritmo standard su $\mathcal{G}'$;

3. salvataggio del ciclo minimo $\mathbf{c}'$ individuato per $\mathcal{G}'$;

4. modifica del ciclo $\mathbf{c}'$ per includere nuovamente il nodo rimosso c_r: due segmenti di $\mathbf{c}'$ vengono modificati in modo da avere c_r come nuovo estremo, trasformando $\mathbf{c}'$ in un ciclo del grafo originario $\mathcal{G}$, che sarà il ciclo minimo individuato $\mathbf{c}^*$; si noti che la posizione di c_r nello spazio non cambia.

Nel passo 4, la reintroduzione del nodo c_r può avvenire in diversi modi. E' ovviamente conveniente implementarla in modo da minimizzare la lunghezza del ciclo $\mathbf{c}^*$; per fare ciò, si consideri che:

- è ottimale reintrodurre il nodo c_r tra due nodi consecutivi di $\mathbf{c}'$;
- se c_p e c_s sono i nodi tra i quali si inserisce c_r, il tratto (c_p, c_s) viene sostituito da (c_p, c_r, c_s);
- il ciclo $\mathbf{c}^*$ è quindi più lungo di $\mathbf{c}'$ della quantità $l_r := d(c_p, c_r) + d(c_r, c_s) - d(c_p, c_s)$.

Conviene dunque scegliere la coppia di nodi consecutivi c_p e c_s in $\mathbf{c}'$ per cui l_r è minima. Il processo di reinserimento del nodo c_r è illustrato in Fig.(6.4). Nel caso di rimozione di più nodi $\{c_{r,1}, ..., c_{r,k}\}$, il processo è analogo; il reinserimento avviene di un nodo per volta, con la stessa strategia di prima.

Le simulazioni sono state svolte attraverso le seguenti fasi:

1. rimozione, a turno, di ciascuno dei 30 nodi di $\mathcal{G}$ e valutazione delle performances dell'algoritmo, per decidere quali nodi fosse più vantaggioso rimuovere;
2. ulteriori analisi (con più iterazioni e più trials) su ciascuna coppia formata da 2 dei 5 migliori nodi individuati nella fase precedente, per valutare la coppia ottimale da rimuovere;
3. analisi grafica dei risultati ottenuti con rimozione momentanea della coppia ottimale, per decidere quale terzo nodo rimuovere simultaneamente.

Dopo l'esecuzione dei primi due passi, la ricerca con rimozione della coppia (c_{10}, c_{23}) è risultata quella più proficua. Il ciclo rilevato più frequentemente $\mathbf{c}^f$, in Fig.(6.5), ha infatti lunghezza $L_{\mathbf{c}^f} = 424.69$, mentre l'algoritmo standard trova in media $\overline{L_{\mathbf{c}}} = 426.36$ (Fig.(4.3(c)), $\rho = 0.7$). Il passo 3 consiste nel confronto tra le Fig.(6.5) e (4.2), dove quest'ultima raffigura uno dei cicli minimi noti (di lunghezza pari a 423.74): come terzo nodo da rimuovere, si è deciso di procedere per tentativi tra quelli che si trovano nella zona in cui le due figure differiscono maggiormente, quindi nella porzione destra di $\mathcal{G}$. Dalle simulazioni è risultato ottimale il nodo c_{22}: con la rimozione dei nodi $\{c_{10}, c_{22}, c_{23}\}$, il ciclo che si trova più frequentemente è quello in Fig.(6.6).

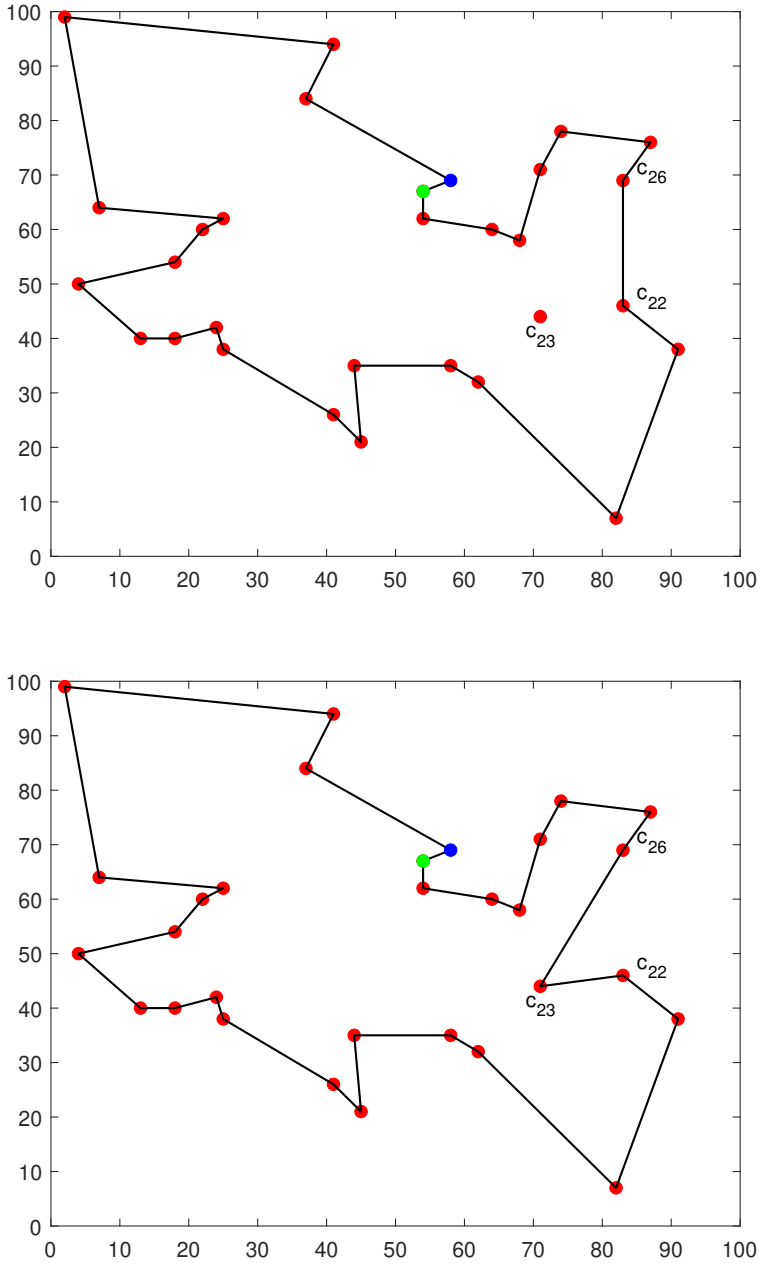


Figura 6.4: Sopra, il ciclo $\mathbf{c}'$, senza il nodo c_{23}. Sotto, il ciclo $\mathbf{c}^*$, con il nodo c_{23}: il segmento $s_{22,26}$ è stato sostituito da $s_{22,23}$ e $s_{23,26}$. La coppia di nodi (c_{22}, c_{26}) era la migliore tra le coppie consecutive di nodi in $\mathbf{c}'$ per la reintroduzione del nodo c_{23}.

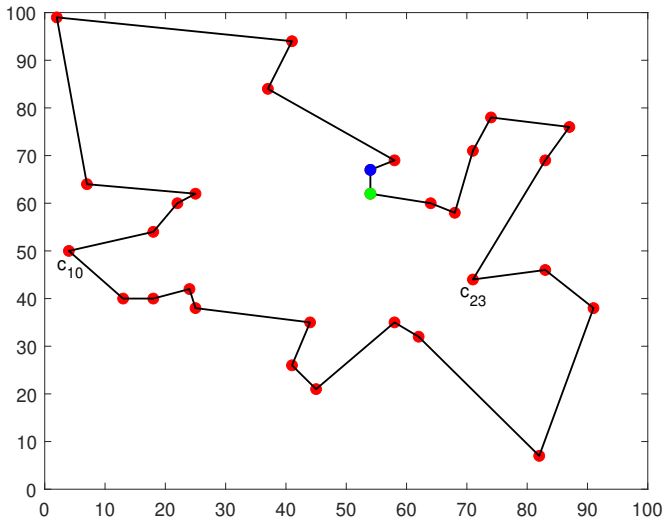


Figura 6.5: Ciclo $\mathbf{c}^f$, il più frequente individuato con rimozione momentanea dei nodi $\{c_{10}, c_{23}\}$.

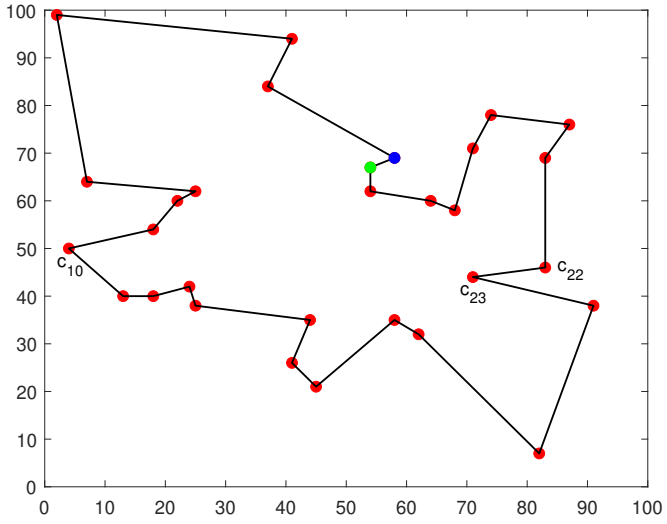


Figura 6.6: Ciclo minimo rilevato più frequentemente con rimozione momentanea dei nodi $\{c_{10}, c_{22}, c_{23}\}$, di lunghezza pari a 424.39. La ricerca è migliorata rispetto alla rimozione dei soli nodi $\{c_{10}, c_{23}\}$, che dava luogo al ciclo $\mathbf{c}^f$ in Fig.(6.5), di lunghezza pari a 424.69. Tuttavia, neanche in questo caso si riesce a ottenere la convergenza a uno dei cicli minimi noti, di lunghezza pari a 423.74.

6.4 Variante con ricerca dei percorsi vicini

In alcuni dei casi in cui l'algoritmo non converge, accade tuttavia che il ciclo rilevato differisca da uno dei cicli minimi noti solo per pochi nodi consecutivi, che vengono collegati in ordine differente. Più precisamente, sia $\mathbf{c}^a := (c_{a_1}, ..., c_{a_n})$ uno dei cicli minimi noti sul grafo $\mathcal{G}$. Può accadere che gli agenti individuino il ciclo $\mathbf{c}^b := (c_{b_1}, ..., c_{b_n})$, dove

$$\exists\, k \in \{1, ..., n\} \mid a_i = b_i \;\forall\, i \in \{1, ..., k\} \cup \{k + v + 1, ..., n\} \tag{6.7}$$

con v "piccolo" (≤ 3). La condizione (6.7) descrive due cicli che differiscono per, al massimo, l'ordine di collegamento dei v nodi nelle posizioni $\{k+1, ..., k+v\}$ dei due cicli. Andando a denotare i cicli $\mathbf{c}^a$ e $\mathbf{c}^b$ tramite la sequenza dei segmenti che li compongono, sono certamente uguali i segmenti

$$\{s_{a_i,a_{i+1}} = s_{b_i,b_{i+1}}\} \;\forall\, i \in \{1, ..., k-1\} \cup \{k + v + 1, ..., n-1\},$$

mentre possono differire i $v + 1$ segmenti restanti. La configurazione descritta è rappresentata in Fig.(6.7).

Poiché i cicli individuati dagli agenti nell'algoritmo standard verificano frequentemente la condizione (6.7) per $v \leq 3$ rispetto a un ciclo minimo noto, può risultare utile indagare i cicli "vicini" a quelli rilevati, dove per "vicini" si intenderanno qui i cammini che differiscono tra loro per l'ordine di al massimo 3 nodi consecutivi ($v = 3$). Se i cicli $\mathbf{c}$ e $\mathbf{c}'$ sono vicini, si scriverà $\mathbf{c} \simeq \mathbf{c}'$. Da un punto di vista computazionale, si noti che, in una generica topologia con n nodi, il numero di percorsi vicini a un ciclo dato si può approssimare a $5n$. Infatti, sia $\mathbf{c}^a := \{c_{a_1}, ..., c_{a_n}\}$; allora:

- le triplette di nodi consecutivi di $\mathbf{c}^a$ sono

$$\begin{cases} (c_{a_i}, c_{a_{i+1}}, c_{a_{i+2}}), \;\; i \in \{1, ..., n-2\} \\ (c_{a_{n-1}}, c_{a_n}, c_{a_1}) \\ (c_{a_n}, c_{a_1}, c_{a_2}), \end{cases} \tag{6.8}$$

  quindi in numero pari a n;

- ciascuna di queste triplette può essere ordinata in $3! = 6$ modi totali, quindi in 5 modi diversi dall'ordine che ha in (6.8).

In realtà, molti dei cicli così calcolati coincidono tra loro: si considerino come esempio i due percorsi che si ottengono dai seguenti riordinamenti di $\mathbf{c}^a$:

$$\begin{cases} (a_1, a_2, a_3) \longmapsto (a_1, a_3, a_2) \\ (a_2, a_3, a_4) \longmapsto (a_3, a_2, a_4). \end{cases} \tag{6.9}$$

Per questa ragione, il valore di $5n$ percorsi vicini è un'approssimazione per eccesso; in ogni caso, si tratta certamente di un valore dell'ordine di n. Per implementare quindi una strategia computazionalmente accettabile, si è optato per il calcolo dei cicli vicini ogni

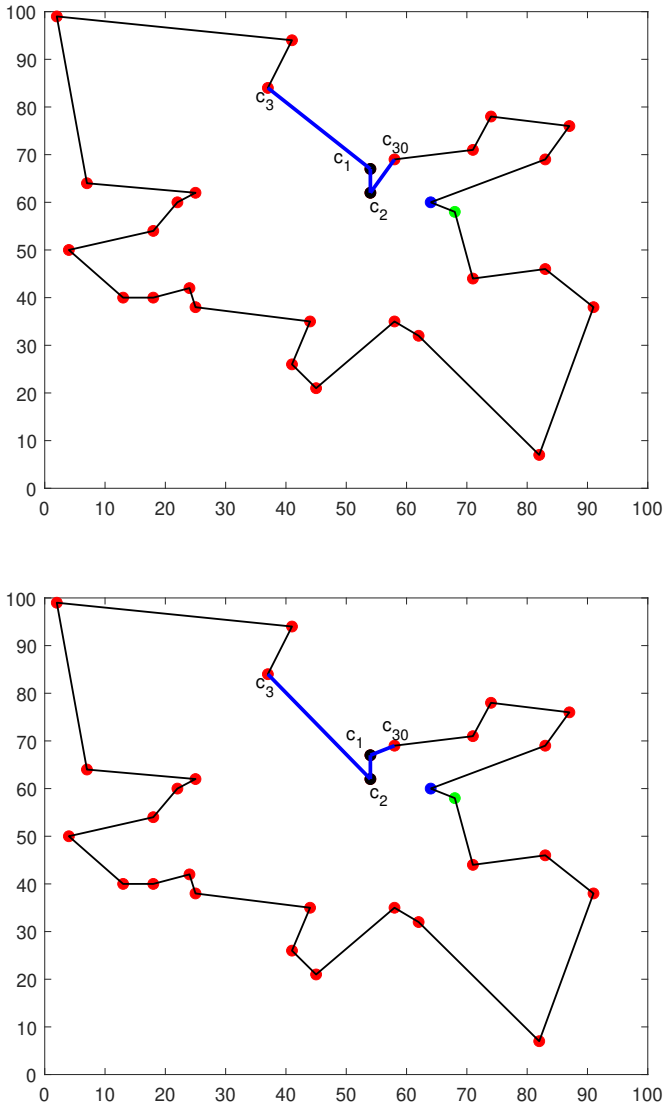


Figura 6.7: Esempio di configurazione per cui è verificata la condizione (6.7) per $v = 2$, $k = 23$. Siano infatti $\mathbf{c}^a$ e $\mathbf{c}^b$, rispettivamente, i cicli nelle figure di sopra e di sotto. Facendo riferimento alla convenzione per cui il primo nodo del percorso è in blu e il secondo in verde, si noti che $a_{23} = b_{23} = 3$, $a_{24} = b_{25} = 1$, $a_{25} = b_{24} = 2$, $a_{26} = b_{26} = 30$. Allora $c_{a_i} = c_{b_i}\ \forall\ i \in \{1, ...,23\} \cup \{26, ...,30\}$: differisce l'ordine dei nodi nelle posizioni {24,25}. I segmenti che differiscono sono i tre rappresentati in blu: per $\mathbf{c}^a$ si ha $(s_{3,1}, s_{1,2}, s_{2,30})$, mentre per $\mathbf{c}^b$ $(s_{3,2}, s_{2,1}, s_{1,30})$.

volta che l'algoritmo individua un ciclo $\mathbf{c}^*$ che migliora il minimo storico $L_{\mathbf{c}^*}$. Simulando su topologia Oliver30, è risultato che tale evento si verifica, nell'algoritmo standard, con una frequenza inferiore all'1% delle iterazioni, dunque sufficientemente bassa. Inoltre, la frequenza diminuisce ulteriormente nel momento in cui i minimi individuati vengono raffintati proprio dall'analisi dei percorsi vicini. Di conseguenza, è stata implementata una ricerca ancora più approfondita, che consiste nell'indagare i "cicli vicini di secondo livello". Il ciclo $\mathbf{c}''$ è detto ciclo vicino di secondo livello del ciclo $\mathbf{c}$ se

$$\exists\ \mathbf{c'}\ |\ \begin{cases} \mathbf{c}'' \simeq \mathbf{c}' \\ \mathbf{c}' \simeq \mathbf{c}. \end{cases}$$

La ricerca dei cicli vicini di primo e secondo livello è strutturata come segue:

1. confronto del percorso minimo dell'iterazione corrente con il ciclo minimo storico $\mathbf{c}^*$;
2. in caso di nuovo ciclo minimo, calcolo dei percorsi vicini di tale ciclo minimo (primo livello);
3. selezione degli *nbest* cicli minimi tra tali percorsi vicini, con *nbest* preimpostato;
4. calcolo dei percorsi vicini per ciascuno degli *nbest* cicli minimi detti (secondo livello);
5. in caso di nuovo minimo, aggiornamento di $\mathbf{c}^*$.

E' stata valutata la convergenza dell'algoritmo con la ricerca dei cicli vicini descritta. In Fig.(6.8) sono presentati i risultati.

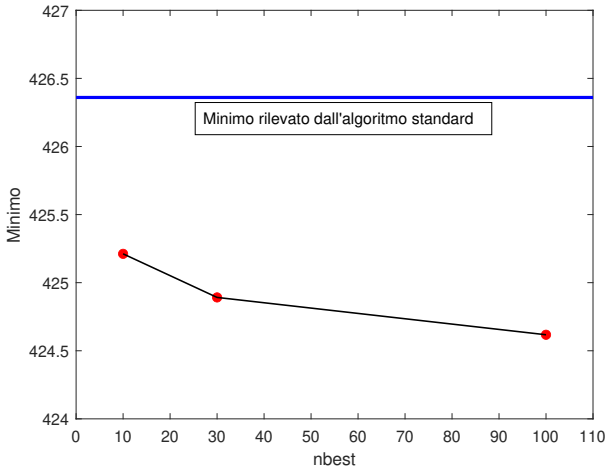


Figura 6.8: Minimo individuato al variare di *nbest*. La linea blu rappresenta il valore medio di $L_{\mathbf{c}^*}$ rilevato dall'algoritmo standard con i parametri standard (Fig.(4.3c), caso $\rho = 0.7$). La strategia si è rivelata efficace nel migliorare l'ottimo individuato dagli agenti; inoltre, indagare un numero maggiore di vicini di secondo livello migliora la convergenza.

Capitolo 7

Applicazione a topologie differenti

Dopo avere esaminato i risultati della ricerca su $\mathcal{G}_4$ e su Oliver30, l'algoritmo è stato applicato ad altre topologie di nodi. Lo scopo di tali analisi è cercare di comprendere in che modo la distribuzione dei nodi nello spazio influisca sull'efficacia di questo metodo di ricerca.

7.1 Topologia Oliver30 con cluster

Una delle topologie proposte è stata ottenuta a partire dalle posizioni dei nodi nel modello Oliver30. L'idea consiste nell'allontanare alcuni dei nodi dal baricentro dei 30 punti, in modo da avere un cluster centrale con la maggior parte dei nodi, mentre i rimanenti vengono distribuiti nelle varie direzioni dello spazio. In particolare, se (x_i, y_i) descrive la posizione del nodo c_i nella topologia Oliver30 classica, le nuove coordinate (x'_i, y'_i) sono:

$$(x'_i, y'_i) = \begin{cases} (x_i + 25(x_i - x_b), y_i + 25(y_i - y_b)) & i \in \{1,4,7,...,28\} \\ (x_i, y_i) & i \notin \{1,4,7,...,28\} \end{cases} \tag{7.1}$$

dove (x_b, y_b) sono le coordinate del baricentro dei 30 nodi originari. La trasformazione (7.1) dà luogo alla topologia rappresentata in Fig.(7.1). E' stata analizzata la variazione del minimo rilevato al variare dei parametri $\{\alpha, \beta, \rho\}$, analogamente a quanto fatto per la topologia Oliver30 classica. La Fig.(7.2) mostra il confronto tra le due ricerche (i risultati della topologia classica sono quelli già mostrati in Fig.(4.3)).

Nonostante nei grafici in Fig.(7.2) si osservi un solo valore di minimo per ogni combinazione di parametri e per ogni topologia, non va dimenticato che il valore mostrato è in realtà la media aritmetica dei risultati di diversi trials, in questo caso 20. Il solo dato della media non fornisce, ovviamente, alcuna informazione sulla dispersione dei risultati. Tuttavia, una ricerca che è capace di individuare minimi vicini a quello reale, seppure poco frequentemente e con numerosi tentativi fallimentari, è diversa da una che rileva

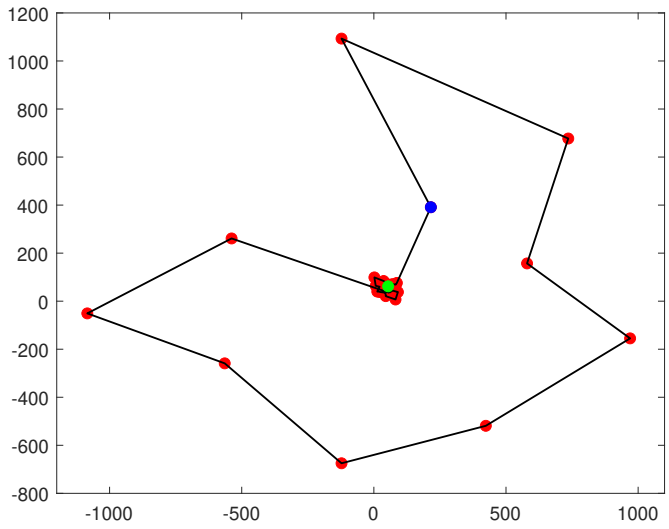


Figura 7.1: Uno dei possibili cicli della topologia descritta, da qui in avanti "Oliver30 con cluster". Si osservano una regione centrale, con i 20 nodi di cui non è stata alterata la posizione, e 10 nodi lontani da essa e tra loro.

sempre valori molto vicini a un dato intermedio. Per indagare l'aspetto della dispersione, sono stati analizzati gli insiemi dei risultati dei vari trials. In particolare, siano

$$\begin{cases} T_{\alpha,\beta,\rho} \in \mathbb{R}_+^{20} \\ T'_{\alpha,\beta,\rho} \in \mathbb{R}_+^{20}, \end{cases} \tag{7.2}$$

rispettivamente, i vettori contenenti i 20 minimi rilevati dagli agenti sulle topologie classica e con cluster, per valori dei parametri pari ad $\{\alpha, \beta, \rho\}$. Siano poi $\sigma_{\alpha,\beta,\rho} \in \mathbb{R}_+$ e $\sigma'_{\alpha,\beta,\rho} \in \mathbb{R}_+$, rispettivamente, le deviazioni standard di $T_{\alpha,\beta,\rho}$ e di $T'_{\alpha,\beta,\rho}$. Al fine di avere dati comparabili tra loro, si sono valutati i coefficienti di variazione

$$\begin{cases} \sigma^*_{\alpha,\beta,\rho} := \frac{\sigma_{\alpha,\beta,\rho}}{\mu_{\alpha,\beta,\rho}} \\ \sigma'^*_{\alpha,\beta,\rho} := \frac{\sigma'_{\alpha,\beta,\rho}}{\mu'_{\alpha,\beta,\rho}}, \end{cases} \tag{7.3}$$

dove $\mu_{\alpha,\beta,\rho}$ e $\mu'_{\alpha,\beta,\rho}$ sono rispettivamente le medie aritmetiche di $T_{\alpha,\beta,\rho}$ e $T'_{\alpha,\beta,\rho}$ in Eq.(7.2). In Fig.(7.3) sono mostrati i pattern di $\sigma^*_{\alpha,\beta,\rho}$ e $\sigma'^*_{\alpha,\beta,\rho}$, da cui emerge che, per alcuni valori dei parametri, i minimi individuati dagli agenti al variare dei trials nel problema con cluster sono molto simili tra loro. In particolare, tale tendenza è più evidente per $\beta > 5$, dove non si verifica invece lo stesso pattern per il problema classico (in cui σ^* torna ad aumentare al crescere di $\beta > 5$).

Tale differenza che si osserva tra le due topologie può essere interpretata riconsiderando il rapporto di preferibilità tra destinazioni definito in (3.1). Per β molto elevato, il termine

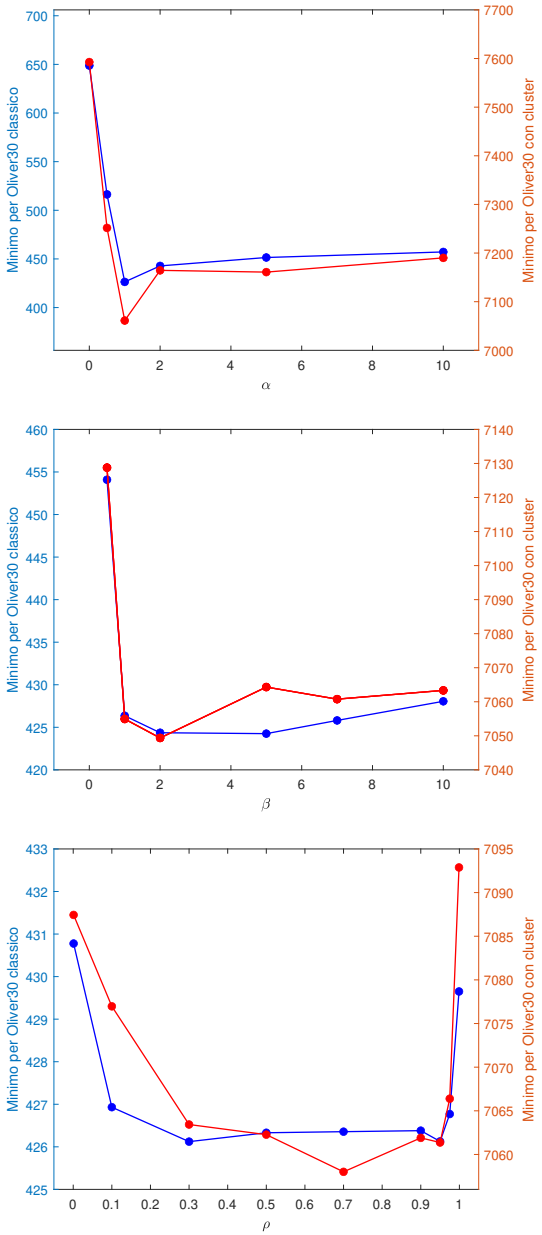


Figura 7.2: Minimo individuato dagli agenti al variare dei parametri $\{\alpha, \beta, \rho\}$ sulle topologie Oliver30 classica (in blu) e con cluster (in rosso), entro $NI_{max} = 5000$ iterazioni. L'analisi è sempre eseguita modificando un solo parametro alla volta e mantenendo gli altri due ai valori standard $\{\alpha = 1, \beta = 1, \rho = 0.7\}$. I pattern osservati sono simili: esistono dei valori intermedi per i quali le ricerche sono ottimizzate, mentre per quelli estremi peggiorano.

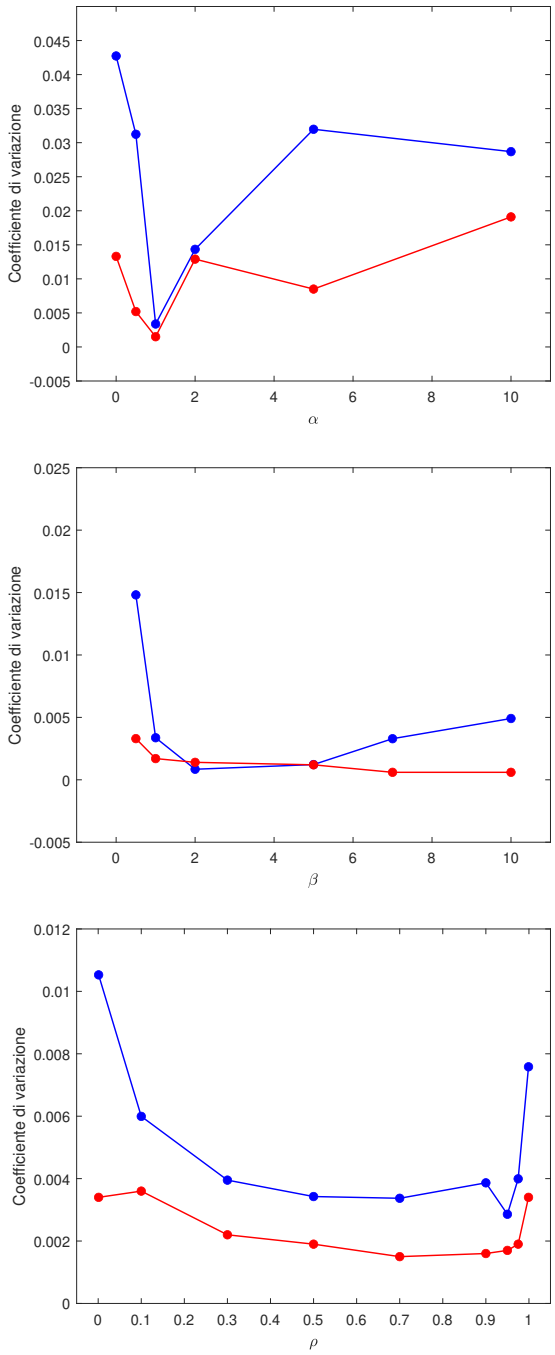


Figura 7.3: Coefficienti di variazione σ^* (in blu) e σ'^* (in rosso) al variare dei parametri $\{\alpha, \beta, \rho\}$. L'andamento complessivo è abbastanza simile nei due problemi, per tutti i coefficienti. Tuttavia, quasi sempre $\sigma^* > \sigma'^*$, in particolare per valori distanti da quelli intermedi per cui i due coefficienti sono ai rispettivi minimi.

del ferormone diventa trascurabile rispetto a quello della visibilità, dunque

$$r_i^{j,k}(t) \simeq \left[\frac{\eta_{i,j}}{\eta_{i,k}}\right]^{\beta}. \tag{7.4}$$

Nel modello con cluster, è frequente che $\eta_{i,j}$ ed $\eta_{i,k}$ siano estremamente diversi, essendoci alcuni nodi molto vicini tra loro e altri sparsi lontano dal cluster; di conseguenza, la scelta del nodo di destinazione è spesso quasi deterministica, molto più frequentemente di quanto ciò non accada per la topologia Oliver30 classica, nella quale i nodi sono distrbuiti nello spazio in modo meno irregolare. La distribuzione delle distanze tra nodi nei due modelli è rappresentata in Fig.(7.4).

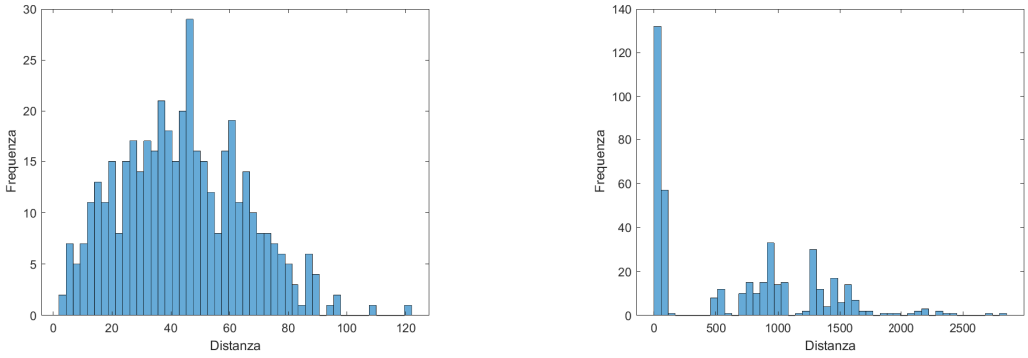


Figura 7.4: Distanze tra i nodi nella topologia Oliver30 classica (a sinistra) e con cluster (a destra). E' evidente come capiti molto più frequentemente che un agente debba scegliere tra due nodi con distanze molto diverse tra loro nel modello con cluster.

7.2 Topologie di griglie esagonali

La topologia Oliver30 con cluster della sezione precedente è chiaramente un modello estremamente irregolare, ancora più dell'Oliver30 classico. In questa sezione saranno presentati invece modelli regolari, come lo era $\mathcal{G}_4$: si tratta di griglie a base esagonale, che ricordano la conformazione degli alveari. E' definito "griglia a base esagonale a $k \in \mathbb{N}_0$ strati" e indicato con $\mathcal{E}^k$ il modello che si costruisce ricorsivamente nel modo che segue:

- $\mathcal{E}^1$ ha come insieme $\{c_1, ..., c_6\}$ dei nodi i vertici di un esagono regolare di lato l;
- $\mathcal{E}^{k\geq 2}$ si ottiene costruendo un esagono regolare su ciascun lato esterno di $\mathcal{E}_{k-1}$.

$\mathcal{E}^3$ è riportato come esempio in Fig.(7.5). Per analizzare le proprietà di $\mathcal{E}^k$, si è introdotta

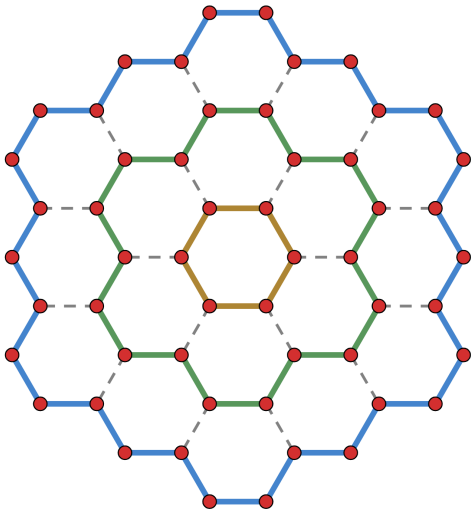


Figura 7.5: L'insieme dei punti rossi rappresenta l'insieme dei nodi di $\mathcal{E}^3$, indicato con $\mathcal{V}^3$ secondo la notazione (7.5). Facendo sempre riferimento alla notazione (7.5), $\mathcal{V}^1$ è l'insieme dei nodi connessi dal percorso marrone e $\mathcal{V}^2$ è l'insieme dei nodi connessi dai percorsi marrone e verde. Di conseguenza, $\overline{\mathcal{V}^1} = \mathcal{V}^1$, $\overline{\mathcal{V}^2}$ è uguale ai soli nodi connessi dal percorso verde e $\overline{\mathcal{V}^3}$ ai soli nodi connessi dal percorso blu.

la seguente notazione:

$$\begin{cases} \mathcal{V}^k \text{ è l'insieme dei nodi di } \mathcal{E}^k, \ \forall\, k \geq 1 \\ \overline{\mathcal{V}^k} := \mathcal{V}^k \setminus \mathcal{V}^{k-1}, \ \forall\, k \geq 2 \\ \overline{\mathcal{V}^1} := \mathcal{V}^1 \\ \mathcal{N}^k := \#(\mathcal{V}^k) \\ \overline{\mathcal{N}^k} := \#(\overline{\mathcal{V}^k}). \end{cases} \tag{7.5}$$

La quantità $\overline{\mathcal{N}^k}$ può essere vista come il numero di nodi della "cornice" k-esima $\overline{\mathcal{V}^k}$ del grafo $\mathcal{E}^k$. Le "cornici" di $\mathcal{E}^3$ sono rappresentate in Fig.(7.5). Da (7.5) si ha

$$\mathcal{N}^k = \sum_{j=1}^{k} \overline{\mathcal{N}^j}, \quad \forall\, k \geq 1, \tag{7.6}$$

che banalmente indica che i nodi di $\mathcal{E}^k$ sono costituiti dall'insieme dei nodi delle cornici che lo compongono. Si noti che

$$\begin{cases} \mathcal{N}^1 = \overline{\mathcal{N}^1} = 6 \\ \overline{\mathcal{N}^k} = \overline{\mathcal{N}^{k-1}} + 12 \quad \forall k \geq 2. \end{cases} \tag{7.7}$$

Dall'Eq. ricorsiva (7.7) si ottiene:

$$\begin{cases} \overline{\mathcal{N}^1} = 6 \\ \overline{\mathcal{N}^k} = 6 + 12(k-1) \quad \forall k \geq 2 \end{cases} \tag{7.8}$$

e quindi, da (7.6) e (7.8),

$$\begin{aligned} \mathcal{N}^k = \sum_{j=1}^{k} 6 + 12(j-1) = \sum_{j=0}^{k-1} 6 + 12j = 6k + 12 \sum_{j=0}^{k-1} j = \\ 6k + 12\left(\frac{(k-1)k}{2}\right) = 6k + 6k^2 - 6k = 6k^2 \quad \forall k \geq 1, \end{aligned} \tag{7.9}$$

dove nella seconda uguaglianza si è effettuata la sostituzione

$$\begin{cases} j' \leftarrow j - 1 \\ j \leftarrow j'. \end{cases}$$

L'Eq.(7.9) mostra che il numero totale di nodi di $\mathcal{E}^k$ è direttamente proporzionale al quadrato del numero k degli strati.

Il seguente teorema consente di avere una stima della lunghezza del ciclo minimo su $\mathcal{E}^k$, $\forall\, k \geq 1$.

Teorema 1. *Sia $L_{\boldsymbol{c}^{*,k}}$ la lunghezza del ciclo minimo $\boldsymbol{c}^{*,k}$ su $\mathcal{E}^k$, $\forall\, k \geq 1$. Vale*

$$l \cdot 6k^2 \leq L_{\boldsymbol{c}^{*,k}} \leq l \cdot (6k^2 + k - 1), \tag{7.10}$$

dove l è la lunghezza del lato dell'esagono.

Dimostrazione. Da (7.9) si ha che $\mathcal{N}^k = 6k^2$. Sia $(c_{a_1}, ..., c_{a_{\mathcal{N}^k}})$ la sequenza di nodi che descrive il ciclo minimo $\mathbf{c}^{*,k}$. I segmenti più brevi di $\mathcal{E}^k$ corrispondono ai lati degli esagoni, dunque

$$\min_{\substack{i,j \leq \mathcal{N}^k \\ i \neq j}} d_{i,j} = l. \tag{7.11}$$

Allora

$$L_{\mathbf{c}^{*,k}} := \left(\sum_{j=1}^{\mathcal{N}^k - 1} d_{a_j, a_{j+1}} \right) + d_{a_{\mathcal{N}^k}, a_1} \geq \left(\sum_{j=1}^{\mathcal{N}^k - 1} l \right) + l = l \cdot \mathcal{N}^k = l \cdot 6k^2, \tag{7.12}$$

ovvero la prima disuguaglianza.

La seconda disuguaglianza viene dimostrata individuando un ciclo $\tilde{\mathbf{c}}^k$ di lunghezza $L_{\tilde{\mathbf{c}}^k}$ uguale al termine di destra di (7.10), $\forall\ k \geq 1$. A quel punto è sufficiente osservare che $L_{\mathbf{c}^{*,k}} \leq L_{\tilde{\mathbf{c}}^k}$. Il caso $k = 1$ è banale, poiché si ha la tesi prendendo $\tilde{\mathbf{c}}^1$ corrispondente all'esagono di $\mathcal{E}_1$. Per $k \geq 2$, invece, si procede per induzione. Il passo base consiste nell'individuare $\tilde{\mathbf{c}}^2$ e dimostrare che si può costruire con una certa procedura a partire da $\tilde{\mathbf{c}}^1$. Si consideri innanzitutto la lunghezza complessiva dei due percorsi marrone e verde in Fig.(7.5): è pari a $l \cdot \mathcal{N}^2 = 24l$, da (7.9). Si osservi ora la Fig.(7.6): il ciclo nero si ottiene a partire dai due percorsi detti, rimuovendo i segmenti rossi tratteggiati e aggiungendo i segmenti neri obliqui: la lunghezza complessiva dei segmenti rimossi è pari a $2l$, quella dei segmenti aggiunti è pari a $3l$, dunque $L_{\tilde{\mathbf{c}}^2} = (24 - 2 + 3)l = 25l$. Si noti che il nodo di partenza non è rilevante. Il passo base è così dimostrato.

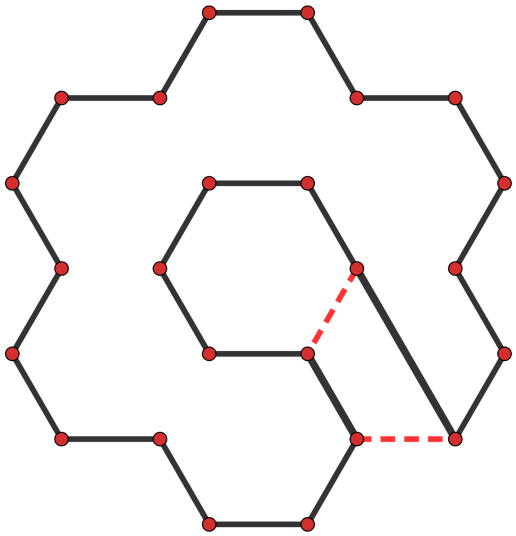


Figura 7.6: La linea nera rappresenta un ciclo su $\mathcal{E}^2$ di lunghezza pari a $25l$, dunque la disuguaglianza di destra di (7.10) è soddisfatta per $k = 2$.

Il passo induttivo consiste nel dimostrare che, se è possibile costruire $\tilde{\mathbf{c}}^k$ a partire da $\tilde{\mathbf{c}}^{k-1}$ per $k > 2$ con la procedura che si è seguita per $k = 2$, in modo che

$$L_{\tilde{\mathbf{c}}^k} = l \cdot (6k^2 + k - 1),$$

allora è anche possibile costruire analogamente $\tilde{\mathbf{c}}^{k+1}$ a partire da $\tilde{\mathbf{c}}^k$, in modo che

$$L_{\tilde{\mathbf{c}}^{k+1}} = l \cdot (6(k+1)^2 + k).$$

Si osservi a questo proposito la Fig.(7.7): i nodi rossi sono quelli del ciclo $\tilde{\mathbf{c}}^k$, il quale, per ipotesi induttiva, è stato costruito a partire da $\tilde{\mathbf{c}}^{k-1}$ con la procedura sopra descritta; la

linea tratteggiata nera corrisponde al tratto che fa parte anche di $\tilde{\mathbf{c}}^{k-1}$ (ne è rappresentata solo una parte). Inoltre, sempre per ipotesi induttiva, $L_{\tilde{\mathbf{c}}^k} = l \cdot (6k^2 + k - 1)$. Si osservi ora l'intero percorso nero; rispetto a $\tilde{\mathbf{c}}^k$:

1. è stata aggiunta la "cornice" $(k+1)$-esima;
2. sono stati aggiunti i due segmenti orizzontali che collegano le due "cornici";
3. sono stati poi rimossi i due segmenti tratteggiati rossi.

Si noti che l'intero percorso nero è un ciclo per $\mathcal{E}^{k+1}$. Il passo 1 consiste nell'aumentare la lunghezza $L_{\tilde{\mathbf{c}}^k}$ del ciclo $\tilde{\mathbf{c}}^k$ di una quantità pari a $\overline{\mathcal{N}^{k+1}} \cdot l = (6 + 12k) \cdot l$, da (7.8); il passo 2 aumenta la lunghezza del ciclo di $3l$; il passo 3 diminuisce la lunghezza del ciclo di $2l$. Si ha perciò:

$$\begin{aligned} L_{\tilde{\mathbf{c}}^{k+1}} &= l \cdot (6k^2 + k - 1) + l \cdot (6 + 12k) + 3l - 2l = \\ &= l \cdot (6k^2 + 13k + 6) = l \cdot (6(k+1)^2 + k), \end{aligned} \tag{7.13}$$

ovvero la tesi. Si noti che la procedura di costruzione di $\tilde{\mathbf{c}}^{k+1}$ a partire da $\tilde{\mathbf{c}}^k$ è sempre possibile: è sufficiente disconnettere la coppia di nodi, nella cornice k-esima, opposta spazialmente a quella che è stata disconnessa per costruire $\tilde{\mathbf{c}}^k$ a partire da $\tilde{\mathbf{c}}^{k-1}$.

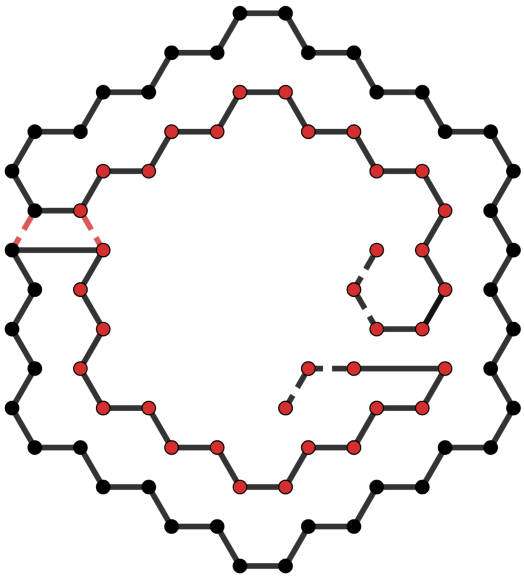


Figura 7.7: Il percorso complessivo in nero rappresenta $\tilde{\mathbf{c}^{k+1}}$ ed è costruito a partire da $\tilde{\mathbf{c}^k}$, ovvero il percorso che congiunge i soli nodi rossi.

□

Il teorema appena dimostrato fornisce una stima dell'errore con cui è possibile calcolare analiticamente $L_{\mathbf{c}^{*,k}}$, $\forall\, k \geq 1$. In particolare:

- per $k = 1$ si ha il valore esatto;
- per $k \geq 2$ l'intervallo in cui si trova $L_{\mathbf{c}^{*,k}}$ ha lunghezza pari a $l \cdot (k-1)$, da (7.10).

Dunque, per $k \geq 2$, l'errore:

- è direttamente proporzionale al numero di strati aggiuntivi rispetto al primo;
- è dello stesso ordine di grandezza di $\sqrt{\mathcal{N}^k}$, da (7.9).

Da un punto di vista sperimentale, è stato valutato il minimo rilevato dagli agenti in alcune griglie $\mathcal{E}^k$, al variare dei parametri α e β. A questo proposito, si consideri la complessità computazionale dell'algoritmo: il costo maggiore è quello relativo al calcolo di tutte le probabilità di destinazione in Eq.(2.2), che viene svolto a ogni iterazione, per ogni step e per ogni agente. Più precisamente, per ogni iterazione si hanno $\mathcal{N}^k$ step, in ciascuno dei quali si considerano M agenti e per ciascuno di questi ultimi vengono calcolate $\mathcal{N}^k$ probabilità di transizione. Per la consueta inizializzazione $M = \mathcal{N}^k$, la procedura descritta ha un costo dell'ordine di $\left(\mathcal{N}^k\right)^3 = 216k^6$ passi per ogni iterazione, dove l'uguaglianza deriva da (7.9). Di conseguenza, se un'analisi su $\mathcal{E}^2$ (24 nodi) può avere un costo computazionale "simile" a quello della ricerca su Oliver30, già per $\mathcal{E}^4$ tale costo viene moltiplicato per

$$\left(\frac{216 \cdot 4^6}{216 \cdot 2^6}\right) = 2^6 = 64. \tag{7.14}$$

Per tale ragione, è estremamente dispendioso eseguire analisi per griglie esagonali con $k \geq 4$ strati.

Per $\mathcal{E}^1$ il minimo reale, corrispondente al percorso esagonale che congiunge i 6 nodi, viene individuato per ogni combinazione di $\{\alpha, \beta\}$, in tutti i 10 trial, entro 1000 iterazioni. Per $\mathcal{E}^2$ l'algoritmo individua cicli minimi della stessa lunghezza per ogni combinazione di parametri. Si tratta di cicli ottenibili per rotazione o per similitudine a partire da quello già visto in Fig.(7.6), utilizzato per dimostrare il Teorema(1). A partire da $\mathcal{E}^3$, la scelta dei parametri inizia ad influire sulla convergenza. Il risultato migliore su $\mathcal{E}^3$ è rappresentato in Fig.(7.8); si noti che la lunghezza del ciclo rilevato è effettivamente nell'intervallo previsto dal Teorema (1), con la disequazione (7.10) che diventa:

$$108 \leq 110.93 \leq 112. \tag{7.15}$$

Nel capitolo 3, si è utilizzato il numero di iterazioni come strumento per valutare la velocità di convergenza su $\mathcal{G}_4$. In questo caso, invece, è stata studiata la frequenza con cui il ciclo minimo trovato viene rilevato al variare di α e β: i risultati sono riportati in Fig.(7.9).

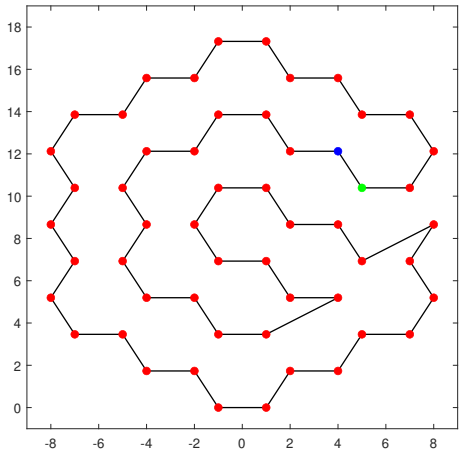


Figura 7.8: Uno dei cicli minimi individuati dal gruppo di agenti su $\mathcal{E}^3$ con lato $l = 2$, di lunghezza pari a 110.93.

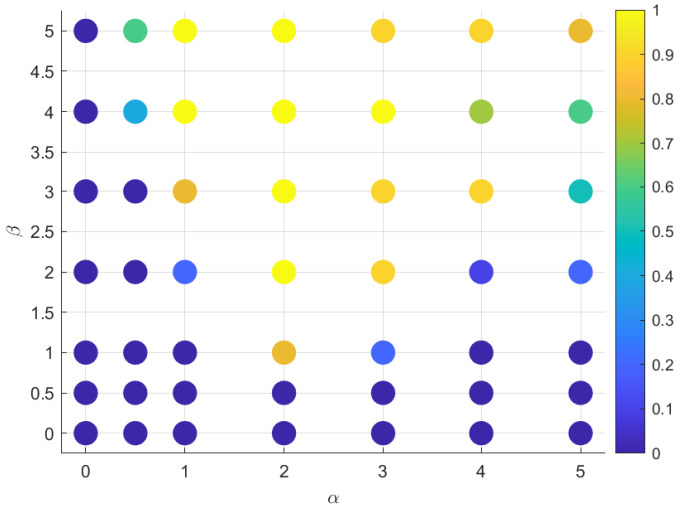


Figura 7.9: Frequenza con cui viene rilevato il risultato migliore dagli agenti al variare di α (in ascissa) e β (in ordinata) in $[0,5] \times [0,5]$, entro $NI_{max} = 1000$ iterazioni e su un totale di 10 trial. La ricerca è molto efficace per $\beta \geq 3$ e, a parità di β, è più efficace per α vicino a 2.

I risultati di convergenza in Fig.(7.9) ricordano quelli in Fig.(3.3) sul grafo $\mathcal{G}_4$, anche se tale analisi era appunto relativa al numero di iterazioni necessarie per la convergenza. In ogni caso, i sottoinsiemi delle combinazioni di parametri che appaiono ottimali per la ricerca sono molto simili:

- la visibilità ha un ruolo predominante; quando è alta, il minimo viene rilevato con frequenza per molti valori diversi di α;
- il ferormone ha una funzione secondaria; solo per β intermedio, scegliere un valore di α prossimo a 2 facilita la convergenza (per $\mathcal{G}_4$ tale valore di riferimento era 1).

Le ragioni sono quelle già elencate nello studio della convergenza su $\mathcal{G}_4$: anche in questo caso, osservando la Fig.(7.8) si nota che i cicli minimi sono costituiti proprio dai segmenti più brevi del grafo (i lati degli esagoni). Di conseguenza, privilegiare di volta in volta i nodi più visibili permette agli agenti di esplorare i percorsi complessivamente più brevi. Si evidenzia comunque una differenza nei risultati di convergenza di $\mathcal{G}_4$ ed $\mathcal{E}^3$: nel primo caso, infatti, il criterio di visibilità era sufficiente a garantire la convergenza a prescindere dal valore di α; nel secondo, invece, non si ha mai convergenza per $\alpha = 0$ (Fig.(7.9)). Alle base di tale differenza potrebbe esserci il numero inferiore di iterazioni eseguite per $\mathcal{E}^3$, dovuto al costo computazionale molto più alto, come spiegato in precedenza.

Capitolo 8

Conclusioni

Complessivamente, la strategia di feedback positivo presentata è utile nell'individuare il minimo reale, o valori comunque prossimi a esso, nel problema del commesso viaggiatore. I due criteri su cui si fonda tale tecnica di ricerca, ovvero intensità di ferormone e visibilità tra nodi, hanno un'importanza relativa che varia a seconda di come è strutturata la topologia del modello. In particolare, sulle griglie regolari la visibilità da sola è sufficiente, sempre o quasi, a impostare una ricerca efficace, mentre per distribuzioni di nodi più disomogenee nello spazio occorre sapere impostare combinazioni ottimali dei parametri in gioco. E' necessario inoltre controllare che l'algoritmo non converga a un percorso unico non minimo, situazione che invalida completamente l'esplorazione del dominio spaziale. La compresenza di molteplici agenti sul grafo si è dimostrata fondamentale ai fini della convergenza. Nei casi in cui la ricerca standard non è efficace, alcune modifiche all'algoritmo si sono rivelate capaci di migliorare i risultati: si tratta dell'introduzione sul grafo di agenti elitari, della rimozione temporanea di nodi e dell'analisi dei percorsi vicini. Al contrario, strategie che consistevano nell'utilizzare un termine di inerzia locale o una componente di movimento totalmente casuale hanno dato esiti controproducenti. Infine, lo studio di un modello estremamente irregolare ha fornito uno spunto per analizzare la correlazione tra distribuzione dei nodi dello spazio e dispersione dei minimi individuati.

Capitolo 9

Appendice

Sono qui riportati i codici Matlab relativi ad alcune delle simulazioni.

```
%% ALGORITMO STANDARD
% TOPOLOGIA: GRIGLIA

lg = 4; % lato della griglia
n = lg^2; % numero di nodi
m = n; % numero di agenti

% DEFINIZIONE DEI NODI
cy = zeros(n,2); % ogni riga conterra' le coordinate di un nodo
c = 1;
% disposizione dei nodi nella griglia
for i=1:lg
    for j=1:lg
        cy(lg*(i-1)+j,:)= [i j];
    end
end
D = squareform(pdist(cy)); % D(i,j) = distanza tra i e j

% INIZIALIZZAZIONE
alfa = 1;
beta = 1;
rho = 0.7;
Q_3 = 100; % quantita' di ferormone da rilasciare
NC_max = 1000; % numero massimo di iterazioni
inf = 16.01; % soglia di ciclo minimo
tau = 10*(ones(n,n)-eye(n,n)); % intensita' iniziale
d_tau = zeros(n,n); % intensita' rilasciata iniziale
sh = sum(sum(D)); % inizializzazione del minimo
b = ones(n,1); % un agente su ogni nodo della griglia
tabu = zeros(m,n+1); % tabu(i,j) contiene il nodo su cui
% si trova l'agente i all'inizio dello step j, l'ultima colonna
% e' uguale alla prima
control = zeros(m,n); % control(i,j) specifica se
% l'agente i ha visitato il nodo j
```

```
k = 1; % indice di riempimento di prima e ultima colonna di tabu

% riempimento di prima e ultima colonna di tabu
for i=1:n
    tabu(k:k+b(i)-1,1) = i;
    tabu(k:k+b(i)-1,n+1) = i;
    control(k:k+b(i)-1,i) = 2; % 2 perche' nodo di partenza
    k = k+b(i);
end

% CORPO DELL'ALGORITMO
for NC = 1:NC_max % ciclo sulle iterazioni
    tabu(:,2:n) = 0; % vengono rimosse le informazioni relative
    % ai nodi visitati, salvo quelli di partenza
    control (control==1) = 0; % come per tabu
    lp = zeros(m,1); % conterra' le lunghezze degli m percorsi

    for step = 1:n % ciclo sugli step
        for j=1:m % ciclo sugli agenti
            if (step<n) % il passo non e' obbligato
                city = tabu(j,step); % nodo dell'agente j
                probg = zeros(n,1); % probabilita' da
                % normalizzare
                for w = 1:n % ciclo sui nodi di destinazione
                    if (control(j,w) == 0) % ovvero se
                    % l'agente j non ha visitato w
                        probg(w) = (tau(city,w)^alfa)/(D(city,w)^beta);
                        % numeratore dell'Eq.(2.2)
                    end
                end
                prob = probg / sum(probg); % normalizzazione
                r = rand;
                p = choice(prob,r); % nodo di destinazione
                tabu(j,step+1) = p; % salvataggio destinazione
                control(j,p) = 1; % l'agente j ha visitato p
            end
            lp(j) = lp(j) + D(tabu(j,step),tabu(j,step+1));
            % aggiornamento della lunghezza del percorso
        end
    end

    for j=1:m % ciclo sugli agenti
        for i=1:n % nuovo ciclo sui nodi visitati
            d_tau(tabu(j,i), tabu(j,i+1)) = ...
                 d_tau(tabu(j,i), tabu(j,i+1)) + Q_3/lp(j);
            % aggiornamento del ferormone rilasciato
            d_tau(tabu(j,i+1), tabu(j,i)) = ...
                d_tau(tabu(j,i), tabu(j,i+1)); % per simmetria
        end
    end

    % AGGIORNAMENTO DELL'INTENSITA' DI FERORMONE
    for ln=1:n
```

```
88          for co=1:n % ciclo su tutti gli elementi di tau
89              if (tau(ln,co) > 1e-50) % per evitare che
90              % diventi un valore che sara' letto come zero
91                  tau(ln,co) = rho*tau(ln,co) + d_tau(ln,co);
92                  % aggiornamento
93              end
94              d_tau(ln,co) = 0; % intensita' rilasciata
95          end
96      end
97
98      % AGGIORNAMENTO DEL MINIMO
99      if (min(lp) < sh)
100         [sh,tp] = min(lp);
101         bt = tabu(tp,:); % salvataggio del ciclo minimo
102     end
103
104     % CONTROLLO DELLA CONDIZIONE DI ARRESTO
105     if (sh<inf)
106         break;
107     end
108
109 end
```

```
1  %% FUNZIONE CHOICE
2  % UTILIZZATA ALLA RIGA 67 DELL'ALGORITMO
3
4  function c = choice(p,r)
5  % r e' un numero random tra 0 e 1
6  % p e' il vettore di probabilita' delle destinazioni
7
8  tot = 0;
9  for i=1:length(p)
10     tot = tot + p(i);
11     if (r<tot)
12         % tale probabilita' dipende
13         % dalla grandezza di p(i)
14         c = i;
15         break;
16     end
17 end}
```

```
1  %% VARIANTE CON RIMOZIONE DI k NODI
2  % SONO CONTENUTE SOLO LE RIGHE NUOVE
3  % RISPETTO ALL'ALGORITMO STANDARD
4
5  % INIZIALIZZAZIONE
6  all_cy = [54 67; 54 62; 37 84; 41 94; 2 99; 7 64; 25 62;
7      22 60; 18 54; 4 50; 13 40; 18 40; 24 42; 25 38; 44 35;
8      41 26; 45 21; 58 35; 62 32; 82 7; 91 38; 83 46; 71 44;
9      64 60; 68 58; 83 69; 87 76; 74 78; 71 71; 58 69];
```

```
% nodi iniziali
rc = [10; 22; 23]; % contiene gli indici dei nodi da rimuovere
rc = sort(rc); % ordinati in modo crescente
lrc = length(rc); % numero dei nodi da rimuovere
n = size(all_cy,1)-lrc; % numero di nodi utilizzati
m = n; % numero di agenti

cy = all_cy(1:rc(1)-1,:);
for i=1:lrc-1
    cy = [cy; all_cy(rc(i)+1:rc(i+1)-1,:)];
end
cy = [cy; all_cy(rc(lrc)+1:(n+lrc),:)];
% contiene i nodi rimanenti

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SEGUE ALGORITMO STANDARD SU cy

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

all_cy_new = [all_cy(rc(1),1) all_cy(rc(1),2)];
for i=2:lrc
    all_cy_new = [all_cy_new; all_cy(rc(i),1) all_cy(rc(i),2)];
end
all_cy_new = [all_cy_new; cy];
%  contiene tutti i nodi ma riordinati
% in modo che i nodi rimossi siano per primi
dcs = squareform(pdist(all_cy_new));
% contiene le distanze tra nodi
dcv = zeros(lrc,n); % dcv(i,j) conterra' la distanza
% tra il nodo i-esimo rimosso e il nodo j-esimo non rimosso
for i=1:lrc
  dcv(i,:) = dcs(i,lrc+1:(n+lrc));
end
dif = zeros(lrc,n);
% dif(i,j) conterra' la lunghezza supplementare
% del percorso con la reintroduzione in posizione j-esima
% del nodo i-esimo rimosso
for i=1:lrc % ciclo sui nodi rimossi
    for j=1:n % ciclo sui nodi non rimossi
        dif(i,j) = dcv(i,bt(j)) + dcv(i,bt(j+1)) - D(bt(j),bt(j+1));
    % differenza tra la lunghezza complessiva
    % dei due nuovi segmenti per reintrodurre in posizione j
    % l'i-esimo nodo rimosso e la lunghezza del segmento
    % che connetteva i nodi non rimossi j e j+1
    end
end
md = zeros(lrc,1); % conterra' i valori
% delle differenze migliori per ciascun nodo rimosso
for i=1:lrc
    md(i) = min(dif(i,:));
    sh = sh + md(i); % lunghezza del ciclo minimo
end
```

```
%% VARIANTE CON PERCORSO UNICO
% SONO CONTENUTE SOLO LE RIGHE NUOVE
% RISPETTO ALL'ALGORITMO STANDARD

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CASO m = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nseq = 0; % contatore di iterazioni consecutive
% con percorsi simili
req_seq = 100; % iterazioni consecutive
% necessarie per percorso unico

% PER OGNI ITERAZIONE NC...
% {

if(NC==1)
    saved = tabu(1,:);
    % alla prima iterazione, saved e' il primo percorso
else
    saved_f = flip(local,2);
    if (tabu(1,:)==saved | ...
            tabu(1,:) == saved_f)
        % il percorso dell'iterazione corrente
        % e' simile a quello precedente
        nseq = nseq + 1;
        % aggiornamento iterazioni consecutive
    else % non sono simili
        nseq = 1; % si ri-inizia a contare da capo
    end
    saved = tabu(1,:); % salvataggio del percorso corrente
end


if (nseq == req_seq)
    % si hanno sufficienti iterazioni consecutive
    % con percorsi simili, dunque percorso unico
    break;
end

% }

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CASO m > 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INIZIALIZZAZIONE
nseq = 0; % contatore di iterazioni consecutive
% con percorsi simili
req_seq = 100; % iterazioni consecutive
% necessarie per percorso unico
eq = zeros(req_seq,1); % conterra' tali iterazioni

```

```
% PER OGNI ITERAZIONE NC ...
% {

nsim = 1; % numero di percorsi simili
control = [repmat(tabu(:,1:n),[2 1]) repmat(flip(tabu(:,1:n),2),[2 1])];
% necessaria a confrontare i percorsi
s1 = find(control(1,:)==1,1);
% indice di inizio del ciclo che parte da 1
% per il primo agente
s3 = find(control(1,:)==1,3);
s3 = s3(3);
% indice di inizio del ciclo che parte da 1
% in senso inverso per il primo agente

for i=2:m % ciclo sugli agenti
    si = find(control(i,:)==1,1);
    % indice di inizio del ciclo che parte
    % da 1 per l'agente i-esimo
    if (control(i,si:si+(n-1)) == control(1,s1:s1+(n-1)) | ...
            control(i,si:si+(n-1)) == control(1,s3:s3+(n-1)))
        % se i cicli sono simili
        nsim = nsim+1;
        % aggiornamento del numero di percorsi simili
    end
end

if (nsim == m)
    % tutti i percorsi dell'iterazione sono simili
    if (nseq == 0)
        % le iterazioni precedenti non avevano percorsi simili
        nseq = 1; % questa e' la prima
        eq(1) = NC; % salvataggio dell'iterazione
    else
        if (eq(nseq) == NC-1)
            % l'iterazione precedente aveva cicli simili
            nseq = nseq+1;
            % aggiornamento del numero di iterazioni
            eq(nseq) = NC;
            % salvataggio dell'iterazione corrente
        else % l'iterazione precedente non aveva cicli simili
            eq = zeros(req_seq,1);
            nseq = 1;
            eq(1) = NC;
            % si ri-inizia da capo a contare quelle consecutive
        end
    end
end

if (nseq == req_seq)
    % si hanno sufficienti iterazioni cosecutive
    % con percorsi simili, dunque percorso unico
    break;
end
```

```
107
108 %}
```

```
1  %% VARIANTE CON INERZIA LOCALE
2  % SONO CONTENUTE SOLO LE RIGHE NUOVE
3  % RISPETTO ALL'ALGORITMO STANDARD
4
5  f = 0.1;
6  % frazione attesa di movimenti secondo inerzia
7  mx = 2*(max(max(D)); % piu' lungo di ogni segmento
8
9  % PER OGNI ITERAZIONE NC...
10 % {
11 %  PER OGNI STEP...
12 %   {
13
14 sw = (rand(m,1)<f);
15 % vettore di zeri e uni con circa percentuale attesa f di uni
16
17 for j=1:m % ciclo sugli agenti
18             if (step<n) % il passo non e' obbligato
19                 if (sw(j) == 1 && step>1)
20                     % movimento secondo inerzia
21                     xd = 2*cy(tabu(j,step),1) - cy(tabu(j,step-1),1);
22                     yd = 2*cy(tabu(j,step),2) - cy(tabu(j,step-1),2);
23                     % destinazioni ideali
24                     closest_cy = [xd yd; cy];
25                     dist_cy = pdist(closest_cy);
26                     dist_cy = dist_cy(1:n);
27                     % distanze della destinazione ideale
28                     % dagli n nodi
29                     for ind = 1:n
30                         if (control(j,ind) > 0)
31                             % l'agente j e' gia' stato in ind
32                             dist_cy(ind) = mx;
33                             % cosi' non sara' selezionabile
34                         end
35                     end
36                     [cl, city_dest] = min(dist_cy);
37                     tabu(j,step+1) = city_dest;
38                     % nodo piu' vicino alla destinazione ideale
39                     control(j,city_dest) = 1;
40                 else
41                     % CODICE STANDARD
```

```
1  %% VARIANTE CON MOVIMENTI CASUALI
2  % SONO CONTENUTE SOLO LE RIGHE NUOVE
3  % RISPETTO ALL'ALGORITMO STANDARD
4
5  f = 0.1; % frazione attesa di movimenti casuali
```

```
6  sl = max(max(D))/2;
7  % lunghezza massima possibile per un passo
8  mx = 2*max(max(D));
9  % piu' lungo di ogni segmento
10
11 % PER OGNI ITERAZIONE NC...
12 % {
13 %  PER OGNI STEP...
14 %   {
15
16 sw = (rand(m,1)<f);
17 % vettore di zeri e uni con circa f % di uni
18
19 for j=1:m % ciclo sugli agenti
20             if (step<n) % il passo non e' obbligato
21                 if (sw(j) == 1)
22                     % movimento casuale
23                     ar = 2*pi*rand;
24                     % angolo casuale tra 0 e 2 pi
25                     lr = sl*rand;
26                     % lunghezza del passo,
27                     % casuale tra 0 e sl
28                     xd = cy(tabu(j,step),1) + lr*cos(ar);
29                     yd = cy(tabu(j,step),2) + lr*sin(ar);
30                     % destinazioni ideali
31                     closest_cy = [xd yd; cy];
32                     dist_cy = pdist(closest_cy);
33                     dist_cy = dist_cy(1:n);
34                     % distanze della destinazione ideale
35                     % dagli n nodi
36                     for ind = 1:n
37                         if (control(j,ind) > 0)
38                             % l'agente j e' gia' stato in ind
39                             dist_cy(ind) = mx;
40                             % cosi' non sara' selezionabile
41                         end
42                     end
43                     [cl, city_dest] = min(dist_cy);
44                     tabu(j,step+1) = city_dest;
45                     % nodo piu' vicino alla destinazione ideale
46                     control(j,city_dest) = 1;
47                 else
48                     % CODICE STANDARD
```

Bibliografia

Jean-Louis Deneubourg and Simon Goss. Collective patterns and decision-making. *Ethology Ecology & Evolution*, 1(4):295–311, 1989.

Jean-Louis Deneubourg, Jacques M Pasteels, and Jean-Claude Verhaeghe. Probabilistic behaviour in ants: a strategy of errors? *Journal of theoretical Biology*, 105(2):259–271, 1983.

Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*, 1992.

Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy. 1991.

Simon Goss, Ralph Beckers, Jean-Louis Deneubourg, Serge Aron, and Jacques M Pasteels. How trail laying and trail following can solve foraging problems for ant colonies. In *Behavioural mechanisms of food selection*, pages 661–678. Springer, 1990.

Martin Grötschel, M Padberg, EL Lawler, JK Lenstra, AHG Rinnooy Kan, and DB Schmoys. The traveling salesman problem, 1985.

JH Holland. Adaptation in natural and artificial systems. ann arbor: University of michigan press, 1975.