Simulated annealing per la gestione di fornitori ed utilizzatori di energia

Michele Roncalli, matricola vr095341

28 aprile 2011

Indice

Elenco delle figure

1 Introduzione

Scopo di questo progetto è l'implementazione dell'algoritmo di *simulated annealing*, un metodo di ricerca locale che si ispira al raffreddamento controllato di un solido dopo un iniziale e repentino innalzamento di temperatura, per portarne la struttura molecolare alla forma cristallina. Il *simulated annealing* opera in maniera simile: per prima cosa genera una soluzione completamente casuale, poi compie una ricerca locale dove i passi *errati* vengono accettati con una probabilità:

$$e^{-\frac{\delta}{T}}$$

che dipende da un parametro T che rappresenta la temperatura che cala in funzione del tempo in base alla funzione, e da δ che è pari alla differenza tra il costo della soluzione nuova e quello della soluzione attuale.

1.1 Un problema concreto

Il problema che si è affrontato è quello di una serie di carichi (o *generatori*) a cui vengono collegati degli utilizzatori (in questo caso, dei *frigoriferi*).

Ogni generatore ha una potenza massima erogabile.

Ogni frigorifero è invece rappresentato da una funzione di utilizzo a scalino, da una tempo di accensione e da una potenza assorbita.

Carichi e utilizzatori vengono collegati tra loro, e il tutto viene rappresentato attraverso una rete di vincoli composta da:

nodi t il nodo i-esimo di questo tipo rappresenta l'istante di accensione dell'i-esimo utilizzatore

nodi g il nodo i-esimo rappresenta a quale generatore si collega l'utilizzatore i-esimo

vincolo unario applicato ad un singolo nodo t, rappresenta l'intervallo di tempo in cui il frigorifero può essere in funzione

vincolo n-ario raggruppa nodi t e g, e rappresenta il generatore a cui i frigoriferi corrispondenti a tali nodi potrebbero collegarsi

La rete viene data in input all'algoritmo, che cerca una soluzione e produce un responso.

2 Implementazione

Il programma è stato sviluppato in java.

La rete di vincoli è rappresentata mediante l'utilizzo di diverse classi: Frigorifero, Generatore, NodoTempo, NodoGeneratore, VincoloUnario, VincoloNario.

La struttura modulare del programma è composta da:

1. l'*input*, che restituisce una rete di vincoli leggendola da un file xml o generandola attraverso dei parametri passati dall'utente;
2. il *solver*, che chiede in ingresso una rete di vincoli, la risolve utilizzando l'algoritmo di *simulated annealing* e produce un responso;

3. l'*output*, che si incarica di produrre un report che rappresenti il risultato ottenuto.

Il tempo viene identificato con le mezz'ore di una giornata: l'istante di accesione va da 0 a 47 (compresi) e la durata di accensione è allo stesso modo indicata dalla quantità di mezz'ore richieste.

2.1 Energia.jar

Il programma viene distribuito mediante un file jar.

Si tratta di un piccolo programma ad interfaccia testuale, a cui si passano i parametri di esecuzione da linea di comando.

Per prima cosa è fondamentale specificare se la rete di vincoli su cui lavorare viene passata mediante un file xml oppure deve essere costruita dal programma stesso. In quest'ultimo caso, c'é la possibilità di specificare i parametri con cui costruire tale rete.

2.1.1 Il file xml

Se si dispone di un file xml che rappresenta la rete, avviare Energia.jar con il parametro

−x | −−xml <*file*>

Nel blocco di codice 1 è mostrato un esempio di file xml accettato dal programma.

Codice 1: Esempio di file xml

```
<?xml version="1.0" encoding="UTF-8"?>
<energia>
        <generatori>
                <generatore>
                        <id>1</id>
                        <maxpower>200</maxpower>
                </generatore>
                <generatore>
                        <id>2</id>
                        <maxpower>100</maxpower>
                </generatore>
        </generatori>
        <frigoriferi>
                <frigorifero>
                        <id>1</id>
                <needtime>4</needtime>
                <watt>100</watt>
                </frigorifero>
                <frigorifero>
                        <id>2</id>
                        <needtime>6</needtime>
                        <watt>100</watt>
                </frigorifero>
```

```
            <frigorifero>
                    <id>3</id>
                    <needtime>10</needtime>
                    <watt>80</watt>
            </frigorifero>
    </frigoriferi>
    <regole>
            <regola>
                    <id_frigo>1</id_frigo>
                    <start>0</start>
                    <end>47</end>
                    <generatori>
                            <id_gen>2</id_gen>
                    </generatori>
            </regola>
            <regola>
                    <id_frigo>2</id_frigo>
                    <start>10</start>
                    <end>37</end>
                    <generatori>
                            <id_gen>2</id_gen>
                    </generatori>
            </regola>
            <regola>
                    <id_frigo>3</id_frigo>
                    <start>1</start>
                    <end>46</end>
                    <generatori>
                            <id_gen>1</id_gen>
                            <id_gen>2</id_gen>
                    </generatori>
            </regola>
    </regole>
</energia>
```

2.1.2 La creazione casuale

La creazione casuale della rete di vincoli avviene specificando il parametro

−c | −−caos

Inoltre, i seguenti parametri vengono utilizzati per guidare la creazione della rete:

−**g** | −−**generatori** numero di generatori;

−f | −−frigorigeri numero di frigoriferi;

−r ripetizioni, ovvero quante volte ripetere l'esperimento. Ad ogni giro viene creata una nuova rete;

−C la funzione di discesa della temperatura scelta. I possibili valori sono: expinv, spezzata, lineare, log o exp;

−n il tipo di passo: random per casuale, aside per incrementale;

−k numero di passi massimi dell'algoritmo;

−−d−num il numeratore della frazione che rappresenta la densità;

−−d−den il denominatore della frazione che rappresenta la densità;

−−max−power−gen la massima potenza erogata dai generatori;

−−max−power−frig la massima potenza assorbita dai frigoriferi;

−−report <file> scrive il report finale su *file*, se non è specificato il report viene stampato nel terminale.

La rete viene creata secondo i parametri stabiliti. La densità è definita come:

d = L/Lmax = L/(F*G)

dove L è il numero di legami, Lmax è il numero massimo di legami, F è il numero di frigoriferi e G il numero di generatori. Il procedimento di creazione casuale del grafo calcola L con la formula:

L = F*G*dnum/dden

dove dnum/dden = densità.

Tuttavia sono stati fissati due limiti: un LOWER BOUND pari al numero di frigoriferi (affinché tutti possano essere collegati ad almeno un generatore) ed un UPPER BOUND pari a F*G*3/4.

La creazione dei legami avviene in tre passi:

1. per ogni frigorifero viene creato un legame con un generatore a caso
2. per ogni generatore, se non ha legami ne crea uno verso un frigorifero a caso
3. crea legami a caso, a patto che il legame non esista già e il generatore non abbia un numero di archi maggiore di 3*F/4

Il procedimento si interrompe non appena si raggiunge il numero di legami L calcolato come sopra. I passi 1 e 2 evitano i casi banali, il passo 3 permette un popolamento non uniforme del grafo ma in linea con i rapporti stabiliti.

2.2 Schedules

Nella definizione di Simulated Annealing, solitamente si dice che l'algoritmo ha un buon comportamento a patto che la temperatura *scenda in modo sufficientemente lento.*

Nella pratica, la discesa del parametro T viene legata ad uno schedule. I più usati sono quello lineare, l'esponenziale o quello logaritmico.

Di seguito, un breve riassunto di quanto ottenuto nei test fatti per questo progetto.

2.2.1 lineare

Lo schedule lineare è definito come

$$T_{k+1} = T_k - c$$

La funzione di discesa è una retta con pendenza negativa. Il parametro c viene impostato dall'utente.

Questo schedule non è stato trovato particolarmente interessante.

2.2.2 spezzata

In questo caso la temperatura scende come in figura 1.

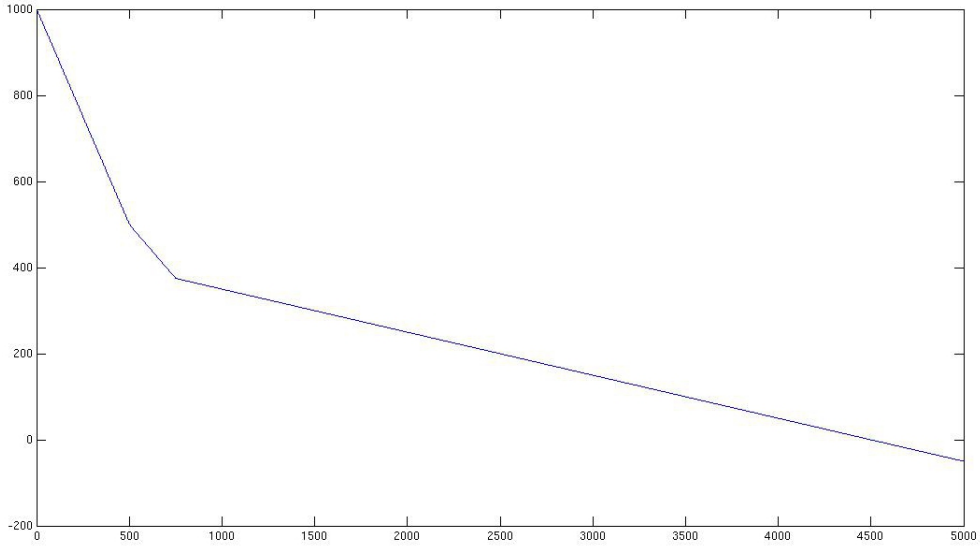

Figura 1: Discesa spezzata.

È stato utilizzato in fase iniziale, per poi venir abbandonato in favore di schedules esponenziali.

2.2.3 esponenziale

Con questo schedule la temperatura segue l'andamento di una esponenziale con esponente negativo, e viene riprodotta in figura 2.

La temperatura crolla in relativamente pochi passi (meno di 200 passi, a fronte di una temperatura iniziale pari a 300), proseguendo poi la ricerca come una ricerca locale classica.

Si è scelto di adottare questo schedule per i sorprendenti risultati ottenuti: in tutti i test effettuati, la discesa esponenziale è quella che più velocemente raggiunge il risultato migliore, a volte impiegando un numero di passi di un ordine di grandezza inferiore rispetto agli altri schedule testati.

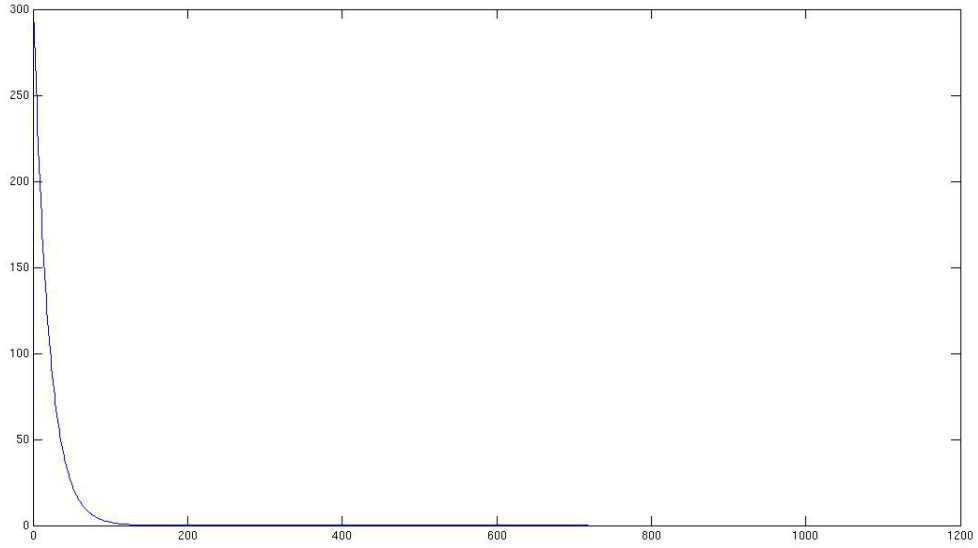

Figura 2: Esponenziale.

2.2.4 logaritmico

Lo schedule logaritmico segue la legge:

$$\frac{1}{\alpha \log(x+2)}$$

dove $\alpha = 0.0048$ per imporre una temperatura iniziale di 300. Esistono in rete diversi articoli dove si dimostra che questo schedule raggiunge l'ottimo. Tuttavia nei test è emerso, al contrario, un comportamento pessimo.

La soluzione del mistero è che lo schedule logaritmico raggiunge l'ottimo in tempo infinito, vista la lentissima discesa del parametro T. Nei test, infatti, la temperatura raggiungeva a malapena un valore di poco inferiore a 20 dopo circa 32000 iterazioni, come si può vedere dalla figura 3, e così il programma prendeva in considerazione troppo spesso passi in direzione *errata*.

Nell'articolo *A comparison of simulated annealing cooling strategies*, di Yaghout Nourani e Bjarne Andresen, si sostiene che:

> As the only existence theorem, it has been proven that for c being greater than or equal to the largest energy barrier in the problem, this schedule will lead the system to the global minimum state in the limit of infinite time. However, due to its asymptotically extremely slow temperature decrease, this schedule is utterly impractical.

Questo schedule è stato implementato, ma non è stato poi usato per lo studio dei problemi descritti nella prossima sezione.

2.2.5 logaritmico inverso

Nella ricerca iniziale del migliore schedule, è stato effettuato un test con una funzione logaritmica sottoposta a traslazione e simmetria rispetto all'asse x. Per maggior chiarezza, si osservi la figura 4. Questo schedule ha

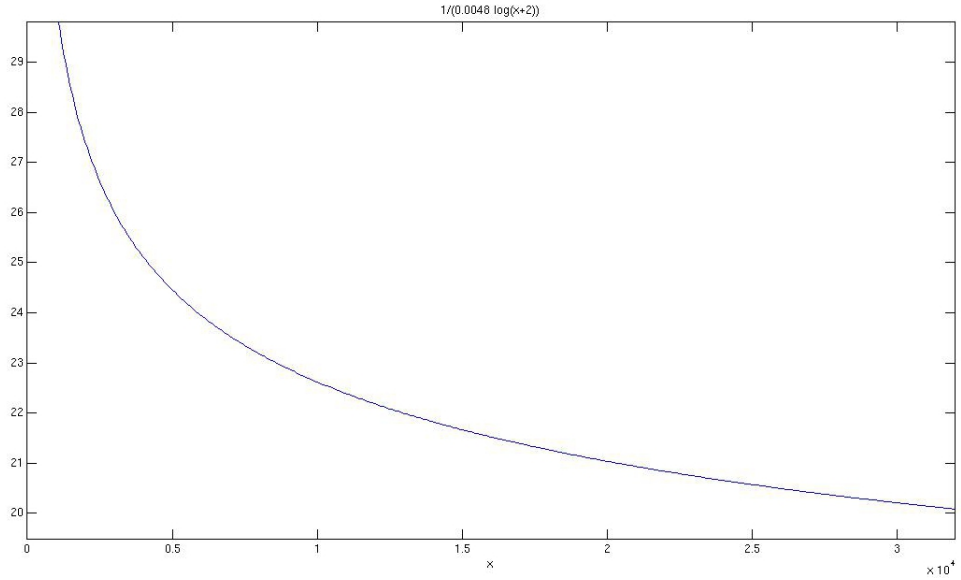

Figura 3: Logaritmico.

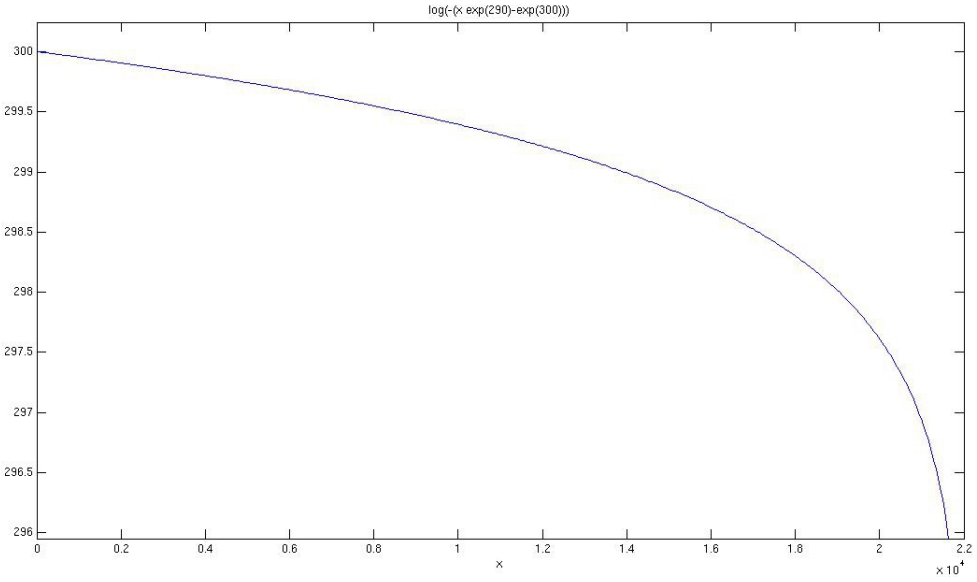

Figura 4: Logaritmico traslata e rovesciata.

prodotto risultati pessimi: T resta prossimo alla temperatura iniziale per poi crollare in poche centinaia di passi alla fine, così praticamente tutto il tempo viene perso in passi *errati* e senza mai identificare una vera e propria direzione in cui cercare.

2.3 Le soluzioni vicine

In un algoritmo di ricerca locale, si parte da una soluzione completa, e si procede esplorando le risorse vicine.

Vista la natura del problema, si sono formulate due diversi tipi di *soluzione vicina*:

- partendo da una soluzione completa, si sceglie un nodo della rete di vincoli, e gli si assegna il valore successivo a quello attuale. In questo caso, una nuova soluzione con costo identico a quello attuale viene sempre accettata, perché ci si trova su un *plateau* e l'algoritmo di simulated annealing prevede di percorrere sempre questa strada;
- come prima, ma il valore per il nodo viene scelto a caso tra quelli possibili. In questo caso, però, una nuova soluzione con costo uguale a quello attuale viene trattata come una soluzione *errata*, quindi accettata solo in base alla probabilità legata a T, poiché in questo caso non si tratta di un *plateau*, ma semplicemente di un'altra soluzione con lo stesso costo che però potrebbe trovarsi molto lontana da quella attuale.

I test preliminari non hanno fatto emergere grosse differenze a livello di prestazioni tra i due approcci.

Gli esperimenti successivi, però, sono stati condotti utilizzando solo la seconda versione, perché così è più semplice spostarsi da un minimo locale.

2.4 La stima della temperatura iniziale

Il valore iniziale della temperatura, detto T_0, è stato inizialmente posto a 300. In alcuni casi è stato inizializzato a 1000.

Tuttavia, per condurre gli esperimenti sui casi in studio, si è preferito implementare un metodo che, data una rete di vincoli, simula n passi iniziali, tiene traccia dei δ ottenuti e in base ad essi stima un valore per T_0 tale che la probabilità di accettare un passo *errato* sia pari a p.

Per il lavoro condotto, p è stato posto a 0.8 e n fissato a 1000 passi.

3 Report

L'algoritmo è stato provato su due diverse classi di esperimenti.

Nel primo caso, si è mantenuto costante il rapporto tra frigoriferi e generatori. Nel secondo, invece, una volta fissato il numero di generatori, sono stati analizzati diversi casi, in ognuno dei quali viene fatto crescere il numero di frigoriferi.

Ad ogni configurazione della rete corrispondono 30 esecuzioni dell'algoritmo. Ad ognuna delle 30 iterazioni viene creata una nuova rete casuale di vincoli. I dati sono stati aggregati e rappresentati attraverso dei grafici prodotti con Matlab.

Per ogni classe di esperimenti viene analizzato l'andamento del miglior costo raggiunto e del numero di passi necessari corrispondente. Si riporta anche l'andamento della soluzione inizale da cui è stata effettuata la ricerca. Infine, lo stesso studio viene ripetuto con differenti valori di densità.

3.1 Caso 1: rapporto costante

Nel primo caso studiato viene mantenuto costante e pari a 2 il rapporto tra frigoriferi e generatori.

3.1.1 passi

Nella tabella in figura 5 vengono raccolti i grafici che rappresentato l'andamento del numero dei passi richiesti per ottenere la soluzione migliore.

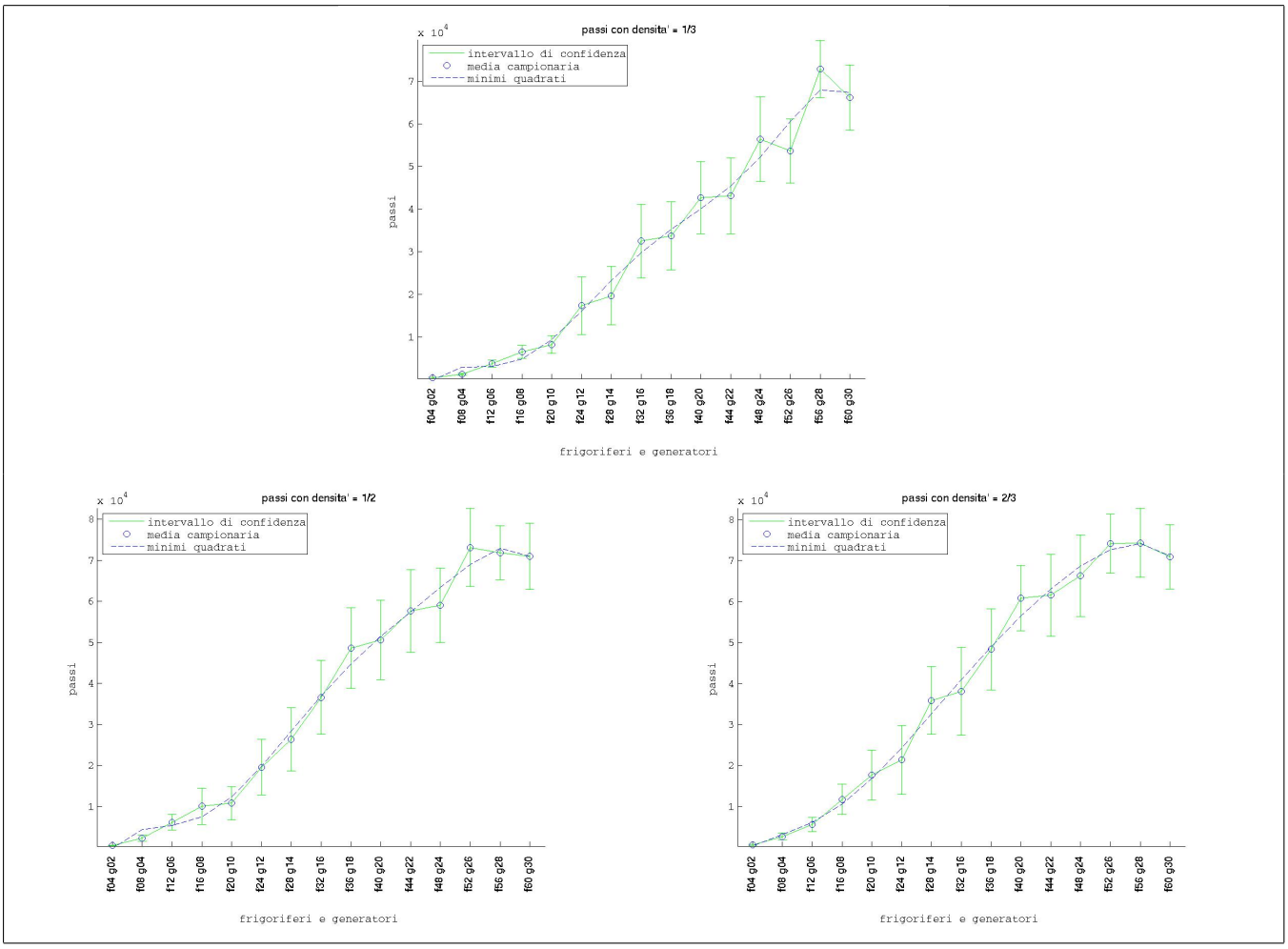

Figura 5: andamento dei passi nel caso 1

Come ci si aspettava, all'aumentare del numero di frigoriferi e generatori, e di conseguenza di nodi della rete, viene compiuto un numero maggiore di passi.

La densità non condiziona in modo evidentemente il numero di passi richiesto, tuttavia si osserva che a valori di densità maggior corrisponde un numero maggiore di passi, che si spiega per il maggior numero di legami presenti e, di conseguenza, un maggior numero di nodi su cui intervenire.

3.1.2 costo

Nella tabella in figura 6 vengono raccolti i grafici che rappresentato l'andamento del costo minimo raggiunto (nelle celle a sinistra) e l'andamento dei corrispondenti costi iniziali (nelle celle a destra).

Per quanto riguarda i grafici sulla sinistra, si osserva che in tutti i casi il costo finale oscilla in un intervallo ampio circa 200.

Qui la densità influenza chiaramente il risultato: nel caso di densità bassa, il costo oscilla tra 800 e 1000, mentre a densità maggiore oscilla tra 400 e 600. Questo comportamento si spiega considerando che la densità è direttamente proporzionale al numero di legami tra frigoriferi e generatori: quando ci sono pochi legami, è più facile che si ottenga un maggior numero di legami non soddisfatti; al contrario, se i legami sono molti, si può ottenere un costo più basso potendo provare un maggior numero di configurazioni.

Nella colonna della destra vengono riportati i grafici che rappresentano l'andamento del costo iniziale, calcolato su una soluzione iniziale generata in modo random dall'algoritmo.

Si può così osservare come l'algoritmo permetta di raggiungere soluzioni sensibilmente migliori di quelle generate a caso: ad esempio nel caso con densità $\frac{2}{3}$, a fronte di soluzioni iniziali che variano tra 40000 e 100000 circa, l'algoritmo ottiene costi tra 300 e 800.

Graficamente questo è rappresentato dalla linea tratteggiata che approssima le soluzioni trovate.

3.2 Caso 2: numero costante di generatori

Il secondo caso analizzato prevede di fissare a 4 il numero di generatori, e provare i diversi casi in cui si ha un numero di frigoriferi che va da 2 a 32.

3.2.1 passi

I grafici che rappresentato l'andamento del numero dei passi richiesti per ottenere la soluzione migliore per questo caso sono raccolti nella tabella di figura 7.

Si possono trarre conclusioni molto simili al caso 1.

In questo caso, l'aumento dei passi è dovuto alla maggiore difficoltà di trovare la soluzione migliore quanto maggiore è il numero di frigoriferi.

Per quanto riguarda la densità, invece, il discorso è del tutto identico al primo caso studiato.

3.2.2 costo

Nella tabella in figura 8 vengono raccolti i grafici che rappresentato l'andamento del costo minimo raggiunto (nelle celle a sinistra) e l'andamento dei corrispondenti costi iniziali (nelle celle a destra).

Il costo finale, come ci si poteva aspettare, aumenta con il crescere del numero di frigoriferi.

Questo comportamento è sensato: quanti più sono i frigoriferi, tanto maggiore sarà il valore della soluzione migliore per via del numero costante di generatori che non può soddisfare richieste via via maggiori.

Di nuovo, la densità influenza i risultati nello stesso modo in cui li influenzava nel caso 1.

Anche per quanto riguarda l'analisi dei grafici della colonna di destra si possono utilizzare le stesse conclusioni tratte nel caso 1.

4 Conclusioni

In questo progetto è stato applicato il *simulated annealing* ad un caso pratico.

Si è così mostrato come un algoritmo di ricerca locale possa essere utilizzato per affrontare un problema reale e quali miglioramenti può portare.

Durante lo sviluppo del progetto si sono esplorati i diversi aspetti del *simulated annealing*, concentrando lo studio sui diversi schedule esistenti e sul relativo comportamento sul caso in analisi.

Inoltre, sono stati sviscerati diversi aspetti secondari: la densità del grafo, la stima della temperatura iniziale, la definizione di soluzione vicina a quella attuale.

I risultati ottenuti sono stati rappresentati attraverso diversi grafici che permettono di mostrare l'andamento dei dati e di condurre una analisi soddisfacente come quella riportata nella presente relazione.

In conclusione, questo progetto si è rivelato interessante per diversi motivi, tuttavia l'aspetto più affascinante è stato quello di trovare applicazione pratica per un algoritmo che, diversamente, avrebbe rischiato di apparire eccessivamente *teorico*.

Figura 6: andamento del costo nel caso 1

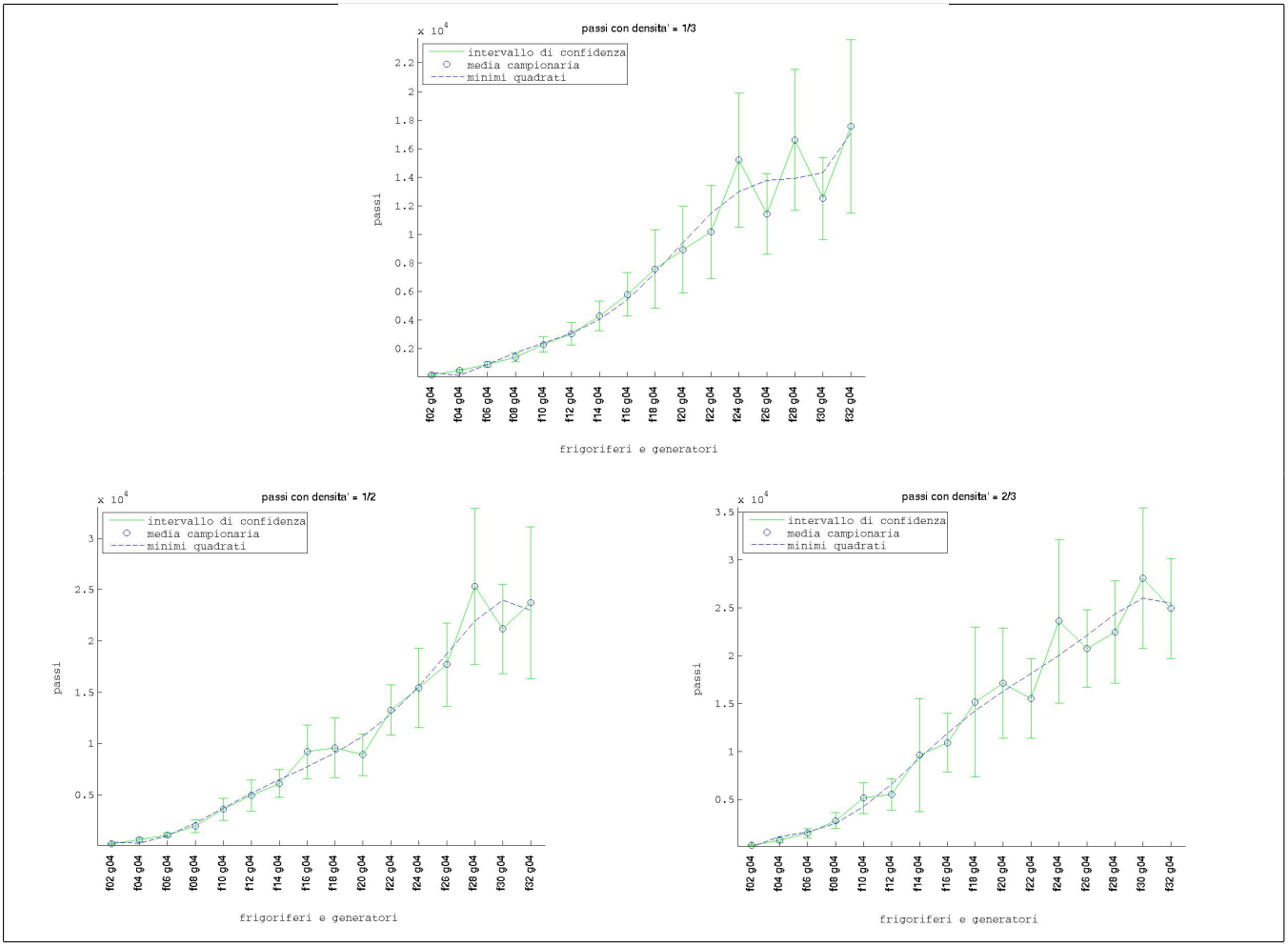

Figura 7: andamento dei passi nel caso 2

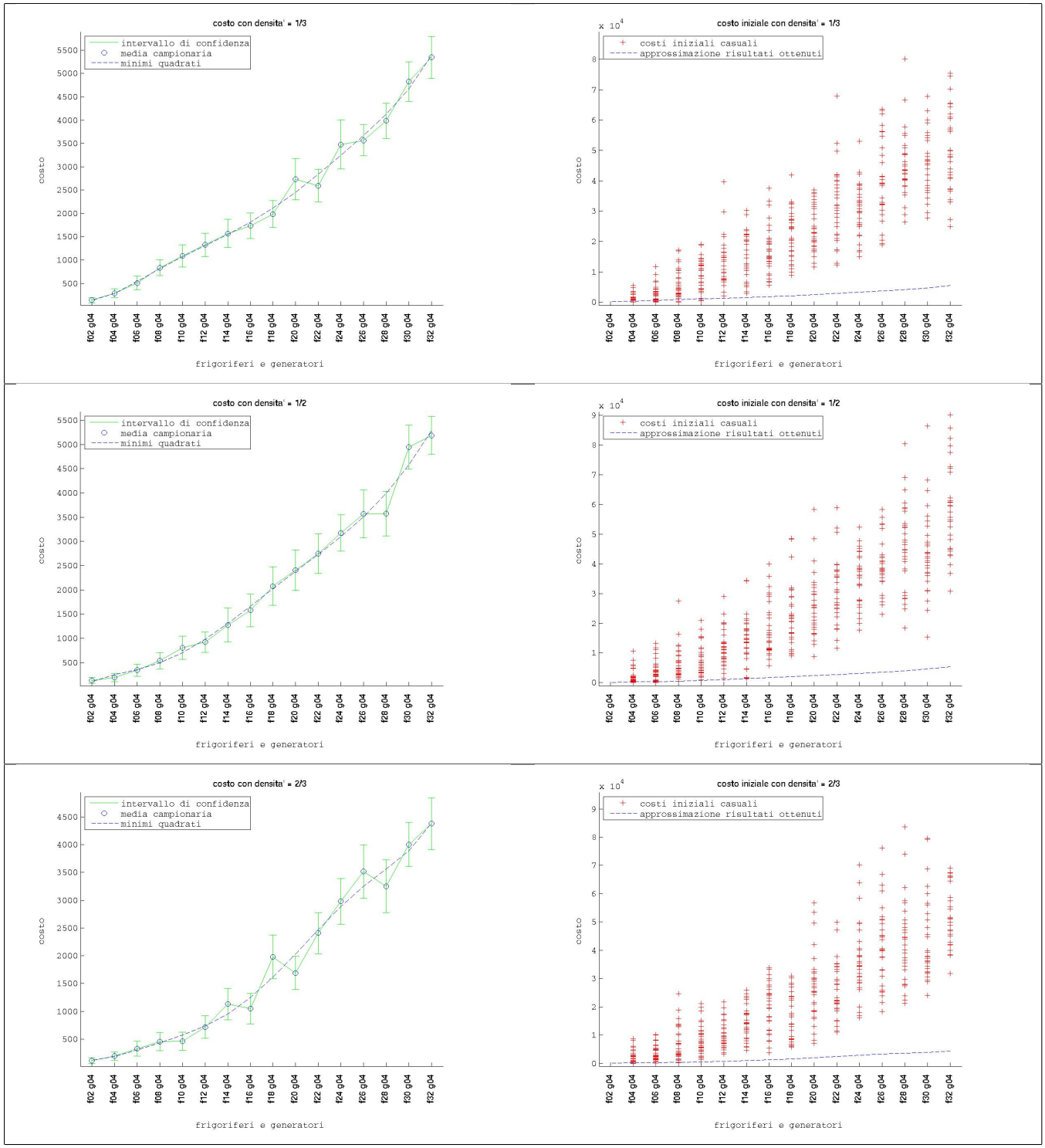

Figura 8: andamento del costo nel caso 2