UNIVERSITÀ DEGLI STUDI DI BOLOGNA
Facoltà di ingegneria
Corso di Laurea in Ingegneria Informatica

Insegnamento: Ricerca Operativa (n° cod. 91008844)

Titolare dell'insegnamento: Silvano Martello (n° cod. 015232)

Candidato all'esame di laurea: Matricola:
Alessandro Trebbi 2148 43952

Titolo della dissertazione:

ALGORITMI EURISTICI PER IL PROBLEMA DELLO UNIT COMMITMENT

Relatore:
Chiar.mo Prof. Ing. Silvano Martello

Correlatori:
Dott. Ing. Alberto Borghetti
Dott. Ing. Andrea Lodi
Prof. Ing. Carlo Alberto Nucci

Parole chiave: *Economic Dispatch Problem*
Unit Commitment Problem
Power Generation
Algoritmi Euristici
Algoritmi Metaeuristici
Tabu Search

I. Lavoro in sintesi

I.1. Introduzione

La domanda ciclica di elettricità nel corso di una giornata richiede alle compagnie erogatrici di pianificare la generazione di potenza. Il corrispondente problema richiede innanzitutto di decidere quali, tra le unità disponibili, accendere, ed in un secondo momento determinare il dispacciamento più economico che soddisfi la richiesta dell'utenza.

Il problema appena descritto è noto come Unit Commitment (UC); è stato oggetto di studi approfonditi negli ultimi 30/35 anni [3]. UC risulta quindi essere un importante sottoproblema di scheduling della produzione. Lo UC Problem (UCP) può essere considerato l'unione di due problemi di ottimizzazione: un problema di ottimizzazione combinatoria e in un problema di ottimizzazione non lineare. L'obiettivo è la determinazione di quali unità generatrici devono essere in servizio durante ogni intervallo dell'orizzonte temporale considerato (un giorno o una settimana), per minimizzare i costi totali, dati da costi di produzione e costi di start-up. La potenza erogata deve soddisfare la domanda dell'utenza con una sufficiente riserva operativa per far fronte ad improvvise emergenze, come picchi di richiesta improvvisa o guasto di una unità. La soluzione deve rispettare vincoli che caratterizzano le singole unità, come vincoli di potenza massima e minima erogabile, minimo tempo di accensione/spegnimento.

Nella formulazione adottata l'orizzonte temporale è suddiviso in intervalli di medesima durata, e la richiesta è assunta costante in ciascun intervallo. La durata tipica di ogni intervallo è assunta pari ad un'ora. La transizione tra due stati diversi di una centrale (ON/OFF) è ammessa esclusivamente all'inizio di ogni intervallo.

La soluzione esatta dello UCP può essere ottenuta ricorrendo ad una enumerazione completa di tutte le possibili combinazioni ammissibili delle unità [10].

Per alcuni, in realtà pochi, problemi di ottimizzazione combinatoria si conoscono algoritmi che trovano la soluzione esatta impiegando un tempo di calcolo che, nel caso peggiore, è una funzione polinomiale della dimensione del problema. Per la maggior parte si conoscono solo algoritmi che trovano la soluzione esatta in un tempo di calcolo che, nel caso peggiore, è una funzione *esponenziale* della dimensione del problema (problemi NP-completi). Lo UCP appartiene a questa classe di problemi.

Un tempo di calcolo esponenziale nella dimensione del problema è inaccettabile per problemi di dimensione reale. L'aumento continuo delle prestazioni dei PC, grazie alle costanti innovazioni tecnologiche nell'ambito dei semiconduttori, non potrà superare il problema [13]. Il ricorso ad algoritmi approssimati diventa dunque assai utile per istanze di grandi dimensioni. Un algoritmo *approssimato* (o *euristico*) è un metodo che cerca una "buona" soluzione con uno sforzo computazionale "accettabile".

Un algoritmo di *ricerca locale* parte da una soluzione ammissibile e, esplorandone un intorno (*neighbourhood*), ne individua altre via via migliori, arrestandosi quando non sono più possibili miglioramenti. Il difetto principale di un simile approccio è che la ricerca può restare "intrappolata" in un minimo locale. Gli algoritmi Tabu Search (TS) superano efficacemente questo svantaggio.

TS è una procedura di ottimizzazione che è stata applicata con successo a numerosi problemi di ottimizzazione combinatoria [27-32]. In questo lavoro verrà proposto un nuovo algoritmo per risolvere lo UCP basato sul metodo TS.

I.2. Algoritmi TS

Viene generata una soluzione ammissibile iniziale, ad esempio ricorrendo ad un algoritmo greedy. Si individua l'intorno della soluzione (*neighbourhood*) costituito da un insieme di soluzioni ammissibili ottenute dalla soluzione corrente mediante applicazione della *mossa*. Quest'ultima definisce come passare da una soluzione ad un'altra. Tra le soluzioni dell'intorno l'algoritmo sceglie poi quel-

la di costo inferiore, indipendentemente dal fatto che porti un miglioramento o meno (*uphill move*). Questo meccanismo intrinsecamente porta a "scappare" dai minimi locali (si veda la figura 0.1), ed implicitamente spinge a cercare un altro minimo vicino a quello appena abbandonato. In pratica, si alterna tra local search, per la ricerca di un ottimo locale, e, una volta che questo è stato trovato, identificazione della miglior soluzione vicina, che viene usata come soluzione di partenza per un nuova fase di ricerca locale.

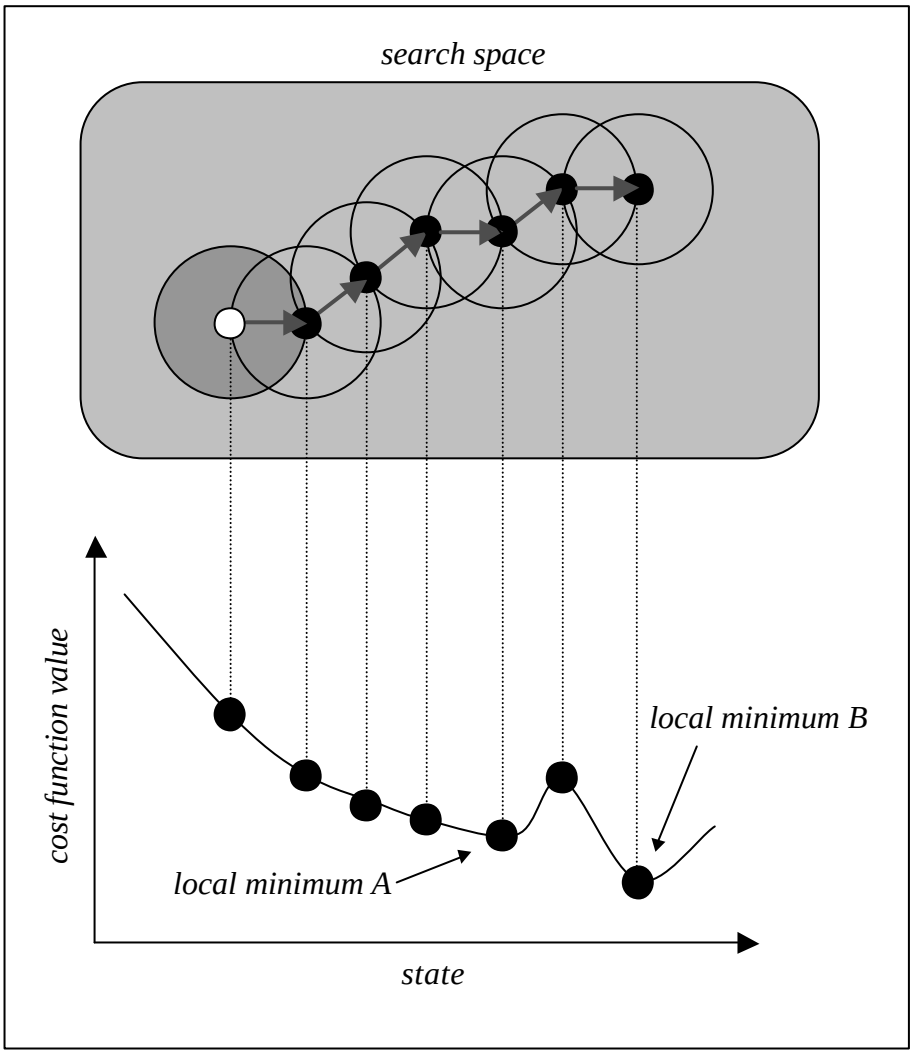


Figura 0.1 – *Neighbourhood search in TS.*

I.2.1 Descrizione di un algoritmo TS

Per descrivere il funzionamento generale di un algoritmo TS si utilizzerà la seguente notazione:

X	insieme delle soluzioni ammissibili dell'istanza;
x	soluzione corrente, $x \in X$;
$x"$	migliore soluzione trovata;
x'	migliore soluzione in un insieme di possibili soluzioni;
$E(x)$	funzione obiettivo della soluzione x;
$N(x)$	neighbourhood di x, insieme dei neighbour ottenuti applicando la mossa, insieme di possibili soluzioni (*trial solutions*);
$S(x)$	sottoinsieme/campione (*sample*) dell'introno di x, $S(x) \in N(x)$;
$SS(x)$	insieme ordinato (*sorted sample*) in accordo con la funzione obiettivo $E(x)$;
TL	tabu list;
AL	aspiration level.

Step 0 Imposta TL come vuoto.

Step 1 Imposta il contatore di iterazioni $K = 0$.
Seleziona una soluzione iniziale $x \in X$, e assegna $x" = x$.

Step 2 Genera un insieme di *trial solutions* $S(x) \in N(x)$ (neighbourhood della soluzione corrente x).
Ordinalo in ordine decrescente per ottenere $SS(x)$.
Sia x' la migliore soluzione possibile di $SS(x)$, ovvero la prima dell'insieme.

Step 3 Se $E(x') > E(x")$, vai allo step 4, altrimenti poni la miglior soluzione $x" = x'$ e vai allo step 4.

Step 4 Esegui il tabu test. Se x' non risulta nella TL, accetta x' come soluzione corrente, imponendo $x = x'$, aggiorna TL e vai allo step 7, altrimenti vai allo step 5.

Step 5 Esegui il test AL. Se soddisfatto, sovrascrivi TL, imponi $x = x'$ e vai allo step 7, altrimenti vai allo step 6.

Step 6 Se è raggiunta la fine di $SS(x)$ vai allo step 7, altrimenti sia x' la prossima soluzione di $SS(x)$ e vai allo step 4.

Step 7 Esegui il test di terminazione. Se il criterio non è soddisfatto imposta $K = K + 1$ e vai allo step 2.

I due principali elementi dell'algoritmo generale appena descritto sono la tabu list e l'aspiration level.

Tabu list

Lo scopo principale della TL è evitare il generarsi di loop che limiterebbero la ricerca nell'intorno di un solo minimo locale. La strategia di vietare, o penalizzare, alcune mosse spinge l'algoritmo a non ripercorrere soluzioni già esaminate.

La scelta di appropriate restrizioni dipende ovviamente dal problema in esame. La decisione di cosa memorizzare si rivela una scelta di progetto cruciale. Normalmente è impossibile, o troppo dispendioso, memorizzare le soluzioni complete corrispondenti alle mosse recenti; in questi casi si memorizzano uno o più attributi di una mossa.

Un secondo problema da affrontare in fase di progetto è relativo alla lunghezza della TL (*tabu tenure*). La TL è gestita in modo da memorizzare le mosse nell'ordine in cui si sono susseguite. Ad ogni iterazione un nuovo elemento è aggiunto alla lista in testa, e l'elemento meno recente viene eliminato. L'effetto di una simile gestione risulta essere che la TL elimina di fatto cicli di lunghezza pari alla dimensione della TL stessa. Sperimentalmente è stato calcolato [11] che la lunghezza della TL che produce buoni risultati spesso aumenta con la dimensione del problema.

Il modo per individuare una buona dimensione della TL è relativamente semplice, e si basa sull'osservazione di risultati sperimentali: una dimensione troppo ridotta porta alla generazione di molti cicli, mentre una dimensione eccessiva porta ad un deterioramento delle soluzioni. Il miglior valore dovrà portare ad un buon compromesso tra questi due estremi. E' stato osservato [33] che in

molte applicazioni la scelta di un range di valori attorno a 7 risulta efficace.

Aspiration criteria

Le indicazioni fornite dalla TL non sono assolute. Se determinate condizioni sono verificate, una determinata mossa può essere accettata anche se risulta "tabu". AL è progettata proprio per questo: per inibire le indicazioni della TL.

In letteratura sono state proposte diverse forme di criteri di aspirazione [27-31]. Il più semplice è sicuramente quello che suggerisce di accettare una soluzione, anche se è tabu, se si riscontra un miglioramento nella funzione obiettivo. Caratteristica comune dei criteri di aspirazione è la capacità di dotare l'algoritmo di maggiore flessibilità indirizzando la ricerca verso quelle soluzioni particolarmente "attraenti".

Termine dell'esecuzione

Vi possono essere diverse condizioni che, se verificate, interrompono l'esecuzione. Normalmente la ricerca termina se il numero di iterazioni eseguite dall'ultimo cambiamento della soluzione migliore supera uno specifico valore, o se il numero di iterazioni totali eccede un dato limite superiore.

I.3. Definizione del modello matematico

I vincoli presi in considerazione si possono suddividere in due gruppi principali: *system constraints* e *unit constraints* (tabella I.1).

system constraints	unit constraints
- load demand *- spinning reserve*	*- generation limits* *- minimum up/down time* *- unit initial status*

Tabella I.1 - *Vincoli del modello matematico.*

I.3.1 Costi operativi di una centrale termica

L'input di una unità termica è generalmente misurato in Btu/h[1], mentre l'output in MW. La curva che lega input all'output è nota con il nome di *heat-rate*. Per ottenere la curva *fuel-cost* è sufficiente convertire l'input da Btu/h a $/h. In tutti i casi pratici, è possibile calcolare i costi operativi ricorrendo ad una funzione quadratica della potenza reale generata [1-3,5-7]:

$$C = a_0 + a_1 P + a_2 P^2 \tag{0.1}$$

I tre coefficienti reali saranno considerati come un input del problema. Il contributo dei costi operativi nella funzione obiettivo risulterà [2,5,7]:

$$\sum_t \sum_i x_{it} \cdot F_i(P_{it}) \tag{0.2}$$

con:

$$F_i(P_{it}) = a_{0i} + a_{1i} P_{it} + a_{2i} P_{it}^{\ 2} \tag{0.3}$$

dove x_{it} è 1 se l'unità i è ON all'istante t; P_{it} è la potenza prodotta dall'unità i nell'unità di tempo t.

I.3.2 Costi di start-up

I costi di start-up sono quei costi che si devono sostenere

[1] *btu = british thermal unit* è una unità di energia corrispondente all'energia che occorre per innalzare di un grado F una lb di acqua. Corrisponde a 1055.056 J. E' l'unità di misura che e' utilizzata in genere per esprimere il potere calorifico (inferiore o superiore) dei combustibili. Quando si ha la curva ingresso-uscita di un impianto in Btu/h in funzione dell'output della centrale in MW (curva determinata con appositi programmi di calcolo termidinamico della centrale, o mediante misure sull'impianto), conoscendo il potere calorifico del combustibile utilizzato (Btu/lb o Btu/kg o Btu/m3, ecc.) e quanto costa una lb o kg o m3 di combustibile, si riesce a determinare anche la curva di costo orario ($/h, Lit/h,ecc.) in funzione dell'output della centrale in MW.

ogni qualvolta si riaccende una centrale. Riportare a regime una unità di produzione richiede un dispendio non trascurabile di risorse. Si è scelto di catturare questo concetto ricorrendo ad una formulazione matematica dei costi di tipo esponenziale[2-3,7]. Tale scelta è stata dettata dalla maggiore precisione che è possibile ottenere rispetto al caso, comunque ampiamente utilizzato, di costi di start-up costanti [4-6].

E' opportuno sottolineare che in letteratura la formulazione esponenziale dei costi di start-up non è omogenea: lavori diversi ricorrono a funzioni esponenziali leggermente diverse. Da un punto di vista concettuale questo fatto non incide in modo rilevante. Tuttavia il confronto di risultati sperimentali risulta inficiato dalla mancanza di omogeneità. In questo lavoro sono state utilizzate diverse formulazioni proprio per poter rendere confrontabile efficacia/efficienza dei diversi algoritmi. Nella formulazione del modello matematico si ricorrerà alla seguente espressione dei costi di start-up:

$$\sum_t \sum_i V_{it} \cdot S_{it} \tag{0.4}$$

con:

$$S_{it} = S_{0i}\left(1 - D_i \cdot e^{-\frac{Toff_{i(t-1)}}{Tdown_i}}\right) + E_i \tag{0.5}$$

dove:
- V_{it} indica lo stato di start-up/shut-down dell'unità i; è 1 se l'unità è stata accesa all'istante t, 0 altrimenti;
- S_{it} start-up cost dell'unità i all'istante t;
- S_{0i} cold start-up cost dell'unità i;
- D_i , E_i coefficienti start-up cost dell'unità i;
- $Tdown_i$ minimo down time dell'unità i;
- $Toff_{it}$ indica da quanto tempo l'unità i è OFF nell'istante.

Per ridurre il numero di variabili in gioco è possibile osservare che l'informazione memorizzata dalla variabile V_{it}, ovvero l'istante in cui una centrale si accende, può essere espressa in modo alternativo:

$$V_{it} = x_{it} \cdot \left(1 - x_{i\cdot(t-1)}\right) \tag{0.6}$$

Ovviamente i costi totali associati allo scheduling saranno dati dalla somma di tutti i costi di start-up e dei costi operativi. La funzione obiettivo del modello matematico risulterà pertanto:

$$\min \sum_t \sum_i x_{it} \cdot F_i(P_{it}) + V_{it} \cdot S_{it} \tag{0.7}$$

I.3.3 Domanda dell'utenza

In ogni intervallo dell'orizzonte temporale il sistema dei generatori deve essere in grado di fornire all'utenza una determinata potenza, dato di input del problema. Per ogni unità di tempo dovrà quindi valere la seguente uguaglianza [1-9]:

$$\sum_i x_{it} \cdot P_{it} = PD_t \qquad \forall t \tag{0.8}$$

dove PD_t è la domanda totale nell'unità di tempo t.

I.3.4 Riserva di potenza

Per dotare il modello della necessaria elasticità è necessario prevedere a priori delle "riserve" di potenza. L'obiettivo è poter far fronte a picchi di richiesta non stimabili e prevedibili a priori. Il sistema di unità produttive deve soddisfare tale richiesta *senza accendere ulteriori centrali*. I vincoli da introdurre saranno pertanto [1-9]:

$$\sum_i x_{it} \cdot P_{i(\max)} \geq PD_t + R_t \qquad \forall t \tag{0.9}$$

dove $P_{i(\max)}$ indica la massima potenza erogabile dall'unità i; R_t è la riserva richiesta all'istante t.

I.3.5 Vincoli di minimo up/down time

Questi vincoli esplicitano il fatto che una centrale non può rimanere accesa/spenta meno di un certo numero di unità di tempo. Matematicamente sarà sufficiente scrivere, per i vincoli di minimo e massimo tempo di spegnimento rispettivamente [4,6]:

$$x_{it} \cdot \left(x_{it} - x_{i(t-1)}\right) \cdot \left(\sum_{j=1}^{Tdown_i} x_{i(t-j)} \right) = 0 \qquad \forall t, i \qquad (0.10)$$

$$x_{i(t-1)} \cdot \left(x_{i(t-1)} - x_{it}\right) \cdot \left(\sum_{j=1}^{Tup_i} x_{i(t-j)} - Tup_i \right) = 0 \qquad \forall t, i \qquad (0.11)$$

dove Tup_i è il minimo up time dell'unità i.

I.3.6 Vincoli sullo stato iniziale

Per ottenere uno scheduling che sia applicabile alla realtà produttiva non si può omettere l'analisi dello stato iniziale delle centrali [2-3]. All'inizio dell'orizzonte temporale di interesse ogni centrale è caratterizzata da uno *stato*, indicante da quante unità di tempo è accesa o spenta. Tali vincoli risultano strettamente legati a quelli di minimo up/down time. La loro espressione matematica non è immediata, e richiede il ricorso a qualche "artificio numerico":

$$F^-\left(Status_i\right) = \left[\mathrm{sgn}\left(Status_i\right) - 1\right] \cdot \left[\sum_{j=1}^{Tdown_i + Status_i} x_{ij} \right] \qquad (0.12)$$

$$F^+\left(Status_i\right) = \left[\mathrm{sgn}\left(Status_i\right) + 1\right] \cdot \left[\sum_{j=1}^{Tup_i - Status_i} (1 - x_{ij}) \right] \qquad (0.13)$$

$$F^+\left(Status_i\right) + F^-\left(Status_i\right) = 0 \qquad (0.14)$$

Ovviamente al termine dell'orizzonte temporale in esame ogni centrale sarà caratterizzata da un nuovo stato, che potrà essere considerato come dato di input per l'orizzonte temporale successivo che si desidera prendere in considerazione.

I.3.7 Vincoli sui generatori

La realtà operativa delle centrali termiche impone di limitare, sia superiormente che inferiormente, la potenza che una centrale è in grado di produrre in ogni istante. Ad esempio, per consentire un corretto ciclo termodinamico dei liquidi nei circuiti idraulici è necessario mantenere determinate temperature, che, come è facile intuire, limitano il range di utilizzo dell'unità.

In generale, quindi, sarà necessario aggiungere al modello matematico visto in precedenza i seguenti vincoli è [1-7]:

$$x_{it} \cdot P_{i(\min)} \le P_{it} \le x_{it} \cdot P_{i(\max)} \qquad \forall i \qquad (0.15)$$

dove $P_{i(\min)}$ e $P_{i(\max)}$ rappresentano rispettivamente la minima e la massima potenza che la centrale i-esima può produrre in una unità di tempo.

I.3.8 Il modello matematico complessivo

$$\min \quad \sum_t \sum_i x_{it} \cdot F_i(P_{it}) + x_{it} \cdot \left(1 - x_{i\cdot(t-1)}\right) \cdot S_{it}$$

$$F_i(P_{it}) = a_{0i} + a_{1i} P_{it} + a_{2i} P_{it}^{\,2}$$

$$S_{it} = S_{0i}\left(1 - D_i \cdot e^{-\frac{Toff_{i(t-1)}}{Tdown_i}}\right) + E_i$$

s.t.

$$\sum_i x_{it} \cdot P_{it} = PD_t \qquad \forall t$$

$$\sum_i x_{it} \cdot P_{i(\max)} \ge PD_t + R_t \qquad \forall t$$

$$x_{it} \cdot \left(x_{it} - x_{i(t-1)}\right) \cdot \left(\sum_{j=1}^{Tdown_i} x_{i(t-j)} \right) = 0 \qquad \forall t, \quad i$$

$$x_{i(t-1)} \cdot \left(x_{i(t-1)} - x_{it}\right) \cdot \left(\sum_{j=1}^{Tup_i} x_{i(t-j)} - Tup_i \right) = 0 \qquad \forall t, i$$

$$x_{it} \cdot P_{i(\min)} \leq P_{it} \qquad \forall t, i$$

$$x_{it} \cdot P_{i(\max)} \geq P_{it} \qquad \forall t, i$$

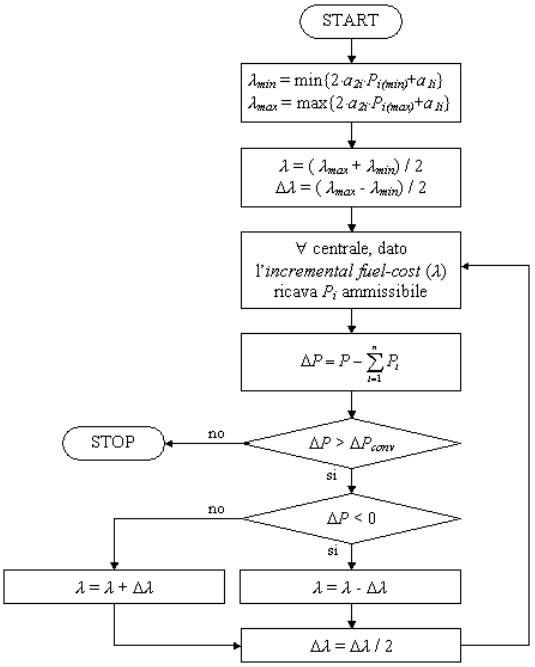
$$\left[\text{sgn}(\, Status_i\,) - 1\right] \cdot \left[\sum_{j=1}^{Tdown_i + Status_i} x_{ij} \right] + \left[\text{sgn}(\, Status_i\,) + 1\right] \cdot \left[\sum_{j=1}^{Tup_i - Status_i} (1 - x_{ij}) \right] = 0 \qquad \forall i$$

$$x_{it} \in \{0,1\} \qquad \forall t, i$$

$$P_{it} \geq 0 \qquad \forall t, i$$

I.4. EDP – Economic Dispatch Problem

L'approccio per individuare iterativamente la soluzione all'EDP [12] è mostrato nello schema seguente.



L'algoritmo determina il valore dell'*incremental fuel-cost* (IFC) "ottimo" utilizzando un metodo di *bisezione*. Individuati gli estremi dell'intervallo di valori ammissibili per l'IFC, se ne calcola il valore medio e si verifica se la potenza associata a tale valore risulta minore, maggiore o sufficientemente vicino alla potenza richiesta. Nei primi due casi si aggiorna l'IFC e si itera nuovamente, nel terzo si termina in quanto è stato individuato l'economic dispatch ottimo con il richiesto grado di precisione.

I.4.1 Un miglioramento proposto

E' opportuno osservare che è assai improbabile che in un determinato istante tutte le centrali accese producano al massimo della loro potenza. E' assai improbabile in quanto nel TS si cerca di individuare nuove soluzioni ammissibili spostandosi nel neighbourhood di una soluzione. Nel fare questo l'algoritmo prova ad accendere/spegnere alcune unità procedendo "a tentativi". E' quindi lecito aspettarsi la maggior parte delle soluzioni costituite da un numero di centrali attive maggiore rispetto al minimo numero possibile. E' assai improbabile che tutte le centrali accese operino alla massima potenza anche in quelle soluzioni "potenzialmente ottime" o molto vicine all'ottimo, a causa della natura quadratica dei costi di produzione: presumibilmente non conviene far produrre una sola centrale alla massima potenza ma due a potenza media.

Queste motivazioni potrebbero essere ritenute sufficienti per sostenere che assai di rado si verifica il caso in cui la potenza richiesta è generata da poche unità termiche operanti al massimo regime di funzionamento. Tuttavia è possibile spingersi oltre, affermando, senza timore di errore, che non è possibile che tutte le centrali operino alla massima potenza. Infatti i vincoli di *spinning reserve* obbligano le unità termiche accese ad operare in modo tale da mantenere, complessivamente, un margine per poter far fronte ad eventuali picchi di richiesta non previsti.

Alla luce di quanto finora discusso è comprensibile come sia preferibile dare priorità diverse ad alcuni valori di IFC: esplorare per primi i valori più probabili e per ultimi quelli meno probabili miglio-

rerebbe l'efficienza dell'algoritmo. Il primo valore considerato è il valore medio dell'intervallo; all'iterazione successiva è verificato il valore associato ad uno degli estremi dell'intervallo. Gli estremi sono quindi valori privilegiati, in quanto l'algoritmo li verifica alla seconda iterazione. Tuttavia, in virtù di quanto appena osservato, è assai improbabile, se non addirittura impossibile, che tali valori risultino quelli corretti per individuare la soluzione ottima all'EDP, e quindi sarebbe opportuno dare loro una priorità molto inferiore. Una diversa inizializzarzione di Δl risolve in modo elegante ed efficiente il problema, riducendo di 1 il numero di iterazioni necessarie per convergere alla soluzione cercata:

$$\Delta l = (l_{max} - l_{min}) / 4 \qquad (0.15)$$

I.5. Generazione di una soluzione iniziale ammissibile

I.5.1 Il problema dello stato iniziale

I vincoli sullo stato iniziale, congiuntamente ai vincoli sul minimo tempo di spegnimento/accensione, vincolano definitivamente una centrale ad essere accesa/spenta in determinate unità di tempo. E' obbligatorio, cioè, tenere accesa/spenta una unità fino a quando risultano soddisfatti i vincoli di minimo up/down time. Le decisioni prese non potranno mai ed in nessun modo essere modificate, in quanto qualsiasi soluzione così cambiata risulterebbe non ammissibile.

Si viene così a delineare un aspetto che caratterizzerà tutta la trattazione nelle prossime pagine. Non tutte le "celle" dello scheduling hanno medesimo ruolo: alcune celle possono contenere degli UNO o degli ZERO indifferentemente, ovvero si può decidere se una unità in un determinato momento è accesa o spenta rispettivamente; altre celle possono contenere uno solo di tali valori, che non si dovrà mai cambiare per mantenere l'ammissibilità della soluzione.

E’ opportuno osservare fin da ora che i valori vincolati si possono determinare senza particolari difficoltà ricorrendo ad una fase di *preprocessing* che deve essere effettuata prima di qualsiasi altra operazione. Tale fase ha esclusivamente il compito di etichettare le celle e assegnare il valore corretto per avere ammissibilità della soluzione. Nel seguito si indicherà con un carattere in grassetto quel valore dello schedule che non si potrà mai modificare (tabella I.2).

0	l’unità è OFF e può essere accesa
0	l’unità è OFF e *non* può essere accesa
1	l’unità è ON e può essere spenta
1	l’unità è ON e *non* può essere spenta

Tabella I.2 - *Notazione adottata*

I.5.2 Un primo semplice algoritmo greedy

Terminata la fase di preprocessing che individua lo stato di alcune centrali non modificabile ai fini della ammissibilità, si procede a completare lo schedule. L’algoritmo greedy adottato a tale scopo è riportato in figura 0.1.

L’algoritmo scandisce tutti gli incroci unità-intervalli. La prima verifica da effettuare è se è già stato assegnato un valore in fase di prepocessing. In caso positivo si passa all’elemento successivo (l’incrocio non è modificabile per i vincoli sullo stato iniziale), altrimenti si decide quale valore assegnare allo schedule. Per determinare tale valore è indispensabile verificare se i vincoli di minimo up/down time sono verificati o meno, ovvero se “sopra” alla cella in esame vi è una sequenza di zeri o di uno pari almeno a *Tdown* o *Tup* rispettivamente. Se questa condizione non è verificata è obbligatorio proseguire nel completamento della sequenza per soddisfare i vincoli del problema. Se invece tale sequenza è già stata formata, l’algoritmo può scegliere se accendere/spegnere la centrale. Per prendere tale decisione è sufficiente verificare se la potenza massima prodotta dalle centrali accese nell’intervallo di tempo soddisfa o meno i vincoli di spinning reserve. Se non è necessario produrre ulte-

riormente si spegne l'unità, altrimenti la si accende.

```
for each unità i do
    for each intervallo t do
        if schedule_ti non è visitato then
            /* verifica se obbligatorio mettere 0 o 1 */
            if sequenza di 1 < Tup_i then schedule_ti = 1;
            if sequenza di 0 < Tdown_i then schedule_ti = 0;
            /* se non obbligatorio mettere 0 o 1 */
            if schedule_ti non è visitato then
                if Σ_i P_i(max) · schedule_ti ≥ PD_t + R_t
                    then schedule_ti = 0;
                    else schedule_ti = 1;
                endif
            endif
        endif
    endfor
endfor
```

Figura 0.1 - *Algoritmo greedy per la determinazione di una soluzione iniziale.*

I.5.3 Spegnere centrali fino ad ottenere l'ammissibilità

Questo secondo approccio, indubbiamente più raffinato per la sua capacità di individuare con più precisione in quale parte dello schedule intervenire, presuppone che tutte le centrali vengano inizialmente accese. L'algoritmo individua quindi per quali intervalli non sono rispettati i vincoli di load demand e spinning reserve, e prova a spegnere centrali in tale intervallo per riportare la soluzione all'ammissibilità.

L'unità di tempo viene determinata osservando la differenza tra minima potenza che le centrali possono produrre (ovviamente è la somma delle $P_{i(\min)}$ delle unità accese) e la potenza richiesta (PD_t). In particolare, l'intervallo selezionato sarà quello per cui tale differenza risulta maggiore. In questo modo si interviene sulla riga dello schedule che maggiormente necessita di intervento per giungere all'ammissibilità.

L'algoritmo esegue poi una scansione da sinistra a destra

delle centrali, fino ad individuarne una che può essere spenta. L'1 associato a tale posizione dello schedule appartiene ad una sequenza verticale di 1, determinata superiormente ed inferiormente da valori diversi. Tra tutti i possibili cambiamenti della sequenza di 1 si individua quello che prevede lo spegnimento della centrale individuata nei passi precedenti, e che porta al miglioramento massimo della funzione obiettivo.

Il "corpo" dell'algoritmo viene ripetuto fino a quando risultano verificati i vincoli di load demand. Con riferimento alla figura 0.2, dove si riporta la pseudocodifica dell'algoritmo completo, si desidera chiarire che il predicato "porta all'ammissibilità" cattura il passaggio da ammissibilità a non ammissibilità (o viceversa) di tutte le righe coinvolte. Il generatore di trial solutions verrà discusso nei prossimi paragrafi.

```
Accendi tutte le unità in tutti gli intervalli;
preprocessing per soddisfare vincoli sullo stato iniziale;
while non ammissibile do
    select t intervallo per cui è massima Σ_i x_ti · P_i(min) - PD_t ;
    i = -1; end = FALSE;
    do
        i ++;
        if x_ti = 1 then
            genera trial solutions sequenza (t,i);
            inizializza bcts e bts; /* Best Trial Solution */
            for each trial solution ts do
                if (t ∈ ts) e (porta all'ammissibilità) then
                    cts = variazione di costo se si applica ts;
                    if cts < bcts then
                        bcts = cts; bts = ts;
                        end = TRUE;
                    endif
                endif
            endfor
        endif
    while end;
endwhile
```

Figura 0.2 - *Algoritmo 2 per la determinazione di una soluzione iniziale.Viene individuata la miglior mossa che porta verso l'ammissibilità.*

I.5.4 Accendere centrali fino ad ottenere l'ammissibilità

Questo terzo approccio è duale rispetto a quello discusso nel paragrafo precedente. Le centrali vengono inizialmente tutte spente. Per quegli intervalli per cui non risultano verificati i vincoli di spinning reserve si provvede ad accendere ulteriori centrali fino all'ammissibilità. Gli intervalli vengono determinati in base alla differenza tra potenza massima erogabile e richiesta dell'utenza (considerando anche la riserva), ovvero tra la somma delle $P_{i(\max)}$ delle unità accese e $PD_t + R_t$. Uno 0 della riga dello schedule appartiene ad una sequenza verticale di 0, determinata superiormente ed inferiormente da valori diversi. Tra tutti i possibili cambiamenti di tale sequenza si individua quella che porta al miglioramento massimo della funzione obiettivo. I passi precedenti vengono ripetuti fino a quando risultano verificati i vincoli di spinning reserve.

I.5.5 Utilità di più euristici

Avere la possibilità di ricorrere a più euristici per determinare una soluzione iniziale ammissibile offre innumerevoli vantaggi.

Un algoritmo di TS può, con opportuni accorgimenti, cominciare a visitare lo spazio delle soluzioni partendo anche da soluzioni non ammissibili. Tuttavia è assolutamente preferibile iniziare l'elaborazione posizionandosi inizialmente in un punto dello spazio delle soluzioni *ammissibili*, per poi muoversi in tale spazio senza abbandonare l'ammissibilità delle soluzioni.

Se capita che un euristico, per la particolare combinazione dei valori dell'istanza numerica in esame, fallisce nel determinare una soluzione ammissibile, è possibile ricorrere ad un altro algoritmo, basato su una logica completamente diversa. Considerando inoltre la possibilità di cambiare l'ordine delle unità, si comprende come di fatto si abbiano a disposizione moltissimi approcci diversi per individuare una soluzione iniziale ammissibile.

Si desidera sottolineare che tutte le istanze reali prese in considerazione hanno sempre portato, alla prima esecuzione, alla de-

terminazione di una soluzione iniziale ammissibile. Solo istanze assolutamente particolari, create appositamente, potrebbero creare problemi di ammissibilità.

Un ulteriore vantaggio è legato alla *diversificazione*. Quando si ritiene che non vi siano buoni ottimi locali "in zona" (un buon indicatore è il non miglioramento da molte iterazioni) è possibile cambiare drasticamente soluzione, posizionandosi in un'altra regione dello spazio delle soluzioni.

I.6. Definizione della mossa: un nuovo approccio metodologico

Uno dei contributi principali di questo lavoro è probabilmente il nuovo approccio proposto per gestire correttamente i vincoli che interessano i generatori (*unit constraints*). Tale approccio è strettamente legato alla metodologia di programmazione object-oriented. Un oggetto che genera trial solutions (o meglio feasible solutions) può sollevare il progettista dalla gestione complessa, e pesante computazionalmente, dell'ammissibilità. Il generatore propone mosse ammissibili, quindi l'ammissibilità non è più un obiettivo da raggiungere, ma è un dato implicito di partenza. L'algoritmo deve limitarsi a selezionare, dall'insieme di mosse generate, quella più opportuna per raggiungere lo scopo prefissato. In altre parole, l'approccio proposto ribalta la sequenza "classica" che prevede in un primo momento la selezione dell'obiettivo da raggiungere (ad esempio accendere una centrale in un determinato intervallo dell'orizzonte temporale) e in un secondo momento l'individuazione del come raggiungerlo (ovvero la determinazione dei cambiamenti da apportare allo schedule per mantenere l'ammissibilità). Questa scelta progettuale ha un impatto sull'intero problema e interessa ogni sua piccola sfaccettatura.

I.6.1 La mossa

Lo schedule risulta formato da sequenze verticali, al limite di lunghezza unitaria, di 0, **0**, 1 e **1**. Alcune di queste sequenze (quelle in grassetto) sono comuni a tutte le soluzioni ammissibili, e quindi non verranno mai modificate. Tutte le altre sequenze possono essere modificate. La definizione della mossa indica come una sequenza può subire modifiche per passare da una soluzione ammissibile ad un'altra. L'insieme delle soluzioni ammissibili ottenibili tramite applicazione della mossa definisce il neighbourhood della soluzione corrente.

In questo lavoro si è voluto definire la mossa in modo tale da generare un intorno della soluzione corrente il più possibile "completo", in modo che l'algoritmo TS potesse esaminare moltissime possibilità ad ogni iterazione e conseguentemente prendere sempre buone decisioni (ottima local search). La mossa risulta pertanto così definita: *tutte* le possibili modifiche di celle *adiacenti* appartenenti alla *medesima sequenza verticale* (non grassetto).

I.6.2 Il generatore di trial solutions

Il generatore di trial solutions ha l'obiettivo di catturare il concetto di mossa introdotto nel precedente paragrafo. Data una sequenza verticale, l'oggetto genera tutti i possibili cambiamenti apportabili a tale sequenza modificando il valore di celle adiacenti.

In generale, è possibile determinare quattro politiche di intervento: è possibile intervenire "in testa" alla sequenza, "in coda" o al centro; la quarta alternativa è cambiare il valore a tutte le celle. Ovviamente queste politiche si articolano in modo diverso a seconda dei casi. Considerando sequenze di 1 (se 0 il discorso è duale), sono stati individuati quattro casi possibili che si possono verificare, dipendenti unicamente dal valore degli estremi della sequenza (una sequenza è individuata da due celle di valore differente).

Una sequenza verticale di 1 è limitata superiormente da un **1** o da uno 0 o da uno **0**. Una sequenza che comincia all'istante iniziale

dell'orizzonte temporale è anch'essa limitata superiormente da uno di tali valori a causa dei vincoli sullo stato iniziale di ogni unità termica. Il limite inferiormente può essere uno 0 o la fine dell'orizzonte temporale. Quest'ultima situazione consente un maggiore grado di libertà e deve quindi essere contemplata.

Si consideri la soluzione ammissibile sotto riportata di un problema caratterizzato da 10 unità. In tale soluzione sono individuabili diverse sequenze verticali; in grigio sono evidenziate quelle sequenze di 1 logicamente diverse in quanto caratterizzate da diversi estremi. Ogni caso, che poi determina le diverse mosse possibili, è stato contrassegnato con una lettera ed esplicitamente riportato a destra.

interval \ *unit*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	**1**	1	**1**	**1**	1	**0**	1	**0**	**0**	0
2	1	1	**1**	1	1	**0**	0	**0**	0	0
3	1	1	**1**	1	1	1	0	0	0	0
4	1	1	1	1	0	1	0	0	1	0
5	1	1	1	0	0	1	0	1	1	1
6	1	1	1	0	0	1	0	1	1	1
7	1	0	1	0	0	1	0	1	1	1
8	0	0	1	0	1	0	0	1	1	1
9	0	0	1	0	1	0	0	1	1	1
10	0	0	1	0	1	0	0	1	1	1
11	0	0	1	0	1	0	0	1	1	1
12	0	1	1	0	1	0	0	1	1	1

case	*0*	*2*	*5*	*7*
A	**1**			
	1		**0**	
	1	**1**	1	
	1	1	1	**0**
	1	1	1	1
	1	1	1	1
	1	1	1	1
	0	1	**0**	1
		1		1
		1		1
		1		1
		1		1
B		**1**		
C			**0**	
D				**0**

Il primo caso, contrassegnato con la lettera A, si verifica quando l'estremo superiore dell'intervallo è un **1**, mentre l'estremo inferiore è uno 0. In tale situazione è possibile estendere lo 0 dell'estremo inferiore verso l'alto, spegnere la centrale per tutti gli intervalli della sequenza, inserire una sequenza di 0 all'interno. Ovviamente questi interventi dovranno essere effettuati rispettando i vincoli di minimo up/down time; i vincoli sullo stato iniziale sono implicitamente verificati in quanto non vengono modificate quelle

celle che contengono valori in grassetto. Considerando valori di *Tup* e *Tdown* rispettivamente pari a 2 e 3, si schematizzano di seguito tutte le possibili trial solutions ottenibili. Sono state separate soluzioni determinabili logicamente in modo diverso.

1	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*
1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0																		

Ogni colonna riporta i nuovi valori delle celle della sequenza di partenza, riportata a sinistra. Nell'esempio sono possibili 18 nuove combinazioni. Sono state evidenziate in grigio scuro quelle celle che cambiano di valore; in grigio chiaro sono contrassegnate le celle che non possono essere modificate in virtù dei vincoli di minimo up/down time. Di seguito si riportano sinteticamente le trial solutions che si possono generare nei restanti casi A, B e C.

1	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*
1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	0	1	1	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	1	1	0	1	0	0
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

0	0
1	
1	
1	
1	
1	
1	
1	
1	
0	

a	*b*	*c*	*d*	*e*	*f*
0	0	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0
1	1	1	1	0	0
1	1	1	1	1	0
1	1	1	1	1	1
1	1	1	1	1	1

g	*h*	*j*	*k*	*l*	*m*
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	0
1	1	1	1	0	0
1	1	1	0	0	0
1	1	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

n	*o*	*p*
1	1	1
1	1	1
0	1	0
0	0	0
0	0	0
1	0	0
1	1	1
1	1	1

q
0
0
0
0
0
0
0
0

0	0
1	
1	
1	
1	
1	
1	
1	
1	

a	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	0	0
1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
1	1	1	1	1	1	1

h
0
0
0
0
0
0
0
0

j	*k*	*l*	*m*	*n*	*o*
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	0
1	1	1	1	0	0
1	1	1	0	0	0
1	1	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

p	*q*	*r*	*s*	*t*	*u*
1	1	1	1	1	1
1	1	1	1	1	1
0	1	1	0	1	0
0	0	1	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
1	1	0	1	0	0
1	1	1	1	1	1

I.7. Algoritmo TS per lo unit commitment

Per risolvere lo UCP devono essere determinate due tipi di variabili. Con riferimento al modello matematico introdotto, le variabili x_{it} che indicano lo stato delle centrali in ogni intervallo sono *intere* (possono assumere valori 0 o 1), mentre quelle che esprimono la potenza generata da ciascuna unità in ogni intervallo dell'orizzonte temporale, P_{it}, sono *continue*. Il problema può quindi essere scomposto in due sottoproblemi: un problema di ottimizzazione combinatoria in x_{it} e in un problema di ottimizzazione non lineare in P_{it}. L'algoritmo TS vero e proprio si occupa della soluzione del primo problema, mentre il secondo è risolto dall'algoritmo per l'EDP. L'algoritmo proposto contiene quindi tre passi principali:

1) generare un set di trial solutions;
2) calcolare il valore della funzione obiettivo risolvendo l'EDP;
3) applicare le procedure caratteristiche del TS per accettare o rifiutare la soluzione in esame.

I.7.1 Selezione della nuova soluzione ammissibile

L'oggetto coordinatore (figura 0.3) individua sullo schedule corrente (associato alla soluzione corrente) una sequenza verticale di 0 o 1. Tale sequenza viene fornita come ingresso al generatore di trial solutions, che, a seconda del caso, genera il set di trial solutions.

Il coordinatore etichetta quali righe individuate dalla sequenza sono ancora ammissibili dopo la complementazione del valore della cella interessata.

Viene analizzata ogni singola trial solution e, se ammissibile, viene calcolato il costo totale associato. Se i costi operativi dopo la complementazione sono già stati calcolati in precedenza si sfrutta l'informazione già esistente, altrimenti si invoca l'algoritmo per l'EDP per ciascuna riga per la quale l'informazione è mancante. I costi operativi così calcolati vengono memorizzati per poter essere sfruttati successivamente.

Se il costo associato alla soluzione risulta il migliore tra quelli calcolati, si memorizza la mossa come best.

Il coordinatore analizza in questo modo tutte le sequenze verticali di 0 o 1. Al termine risulterà memorizzata la mossa appartenente al neighbourhood della soluzione corrente che porta ad una nuova soluzione ammissibile di costo minimo. Tale soluzione diventa la nuova soluzione corrente.

Memorizzare una sola feasible solution è corretto se e solo se si può garantire che la mossa che porta alla miglior soluzione ammissibile del neighbourhood può essere sicuramente accettata. E' indispensabile effettuare il tabu test (e l'aspiration test) subito dopo la fase di verifica dell'ammissibilità. Solo in questo modo si può essere certi che la soluzione migliore può essere accettata, ovvero solo anticipando i suddetti controlli è possibile memorizzare una sola mossa. In figura 0.4 si è riportato l'algoritmo.

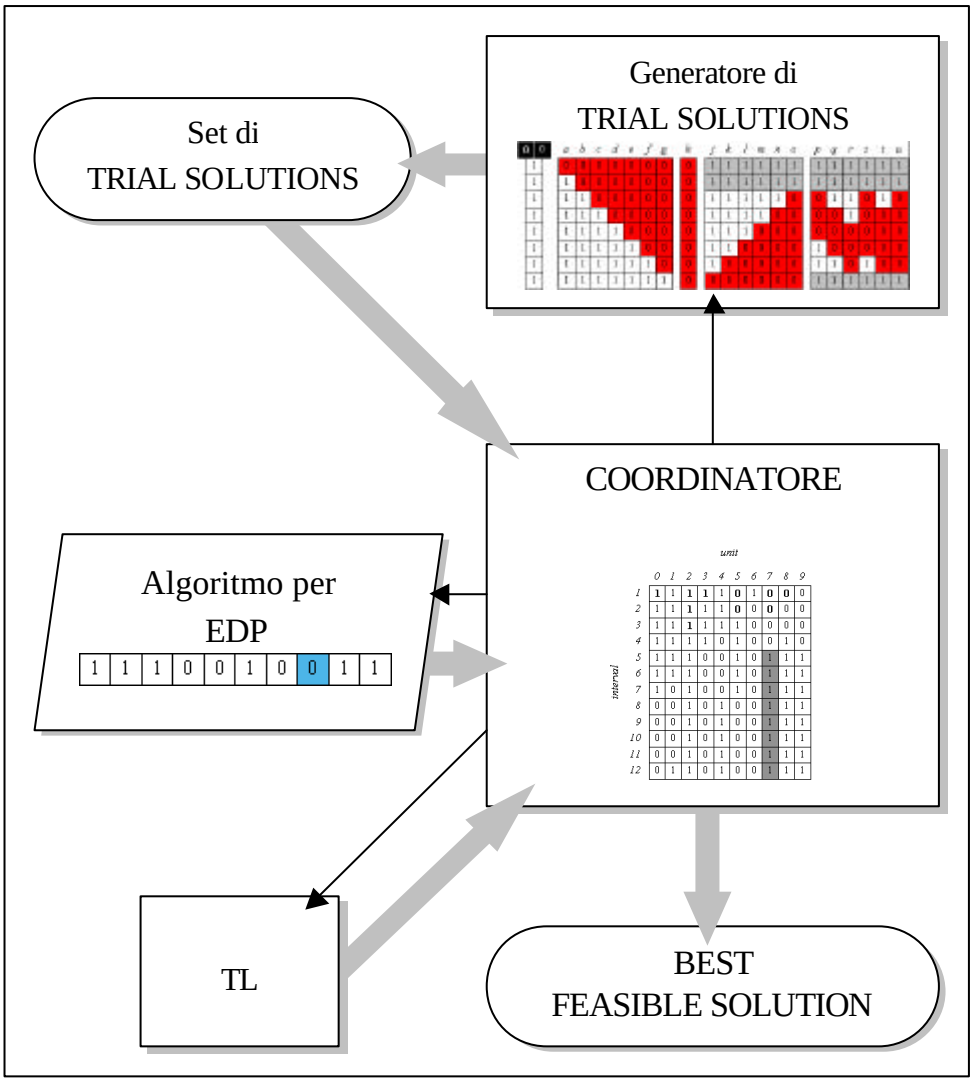


Figura 0. 3 – *Descrizione delle interrelazioni tra i vari componenti software che portano alla determinazione della soluzione ammissibile che sostituirà la corrente.*

I.7.2 L'algoritmo TS completo

L'algoritmo TS, dopo una opportuna fase di inizializzazione delle principali variabili, determina una soluzione iniziale ammissibile eseguendo uno degli euristici descritti, eventualmente dopo aver ordinato le colonne in modo opportuno. La soluzione ammissibile così determinata diventa la soluzione corrente.

```
for each sequenza verticale di 0 o 1 do
    genera trial solutions;
    for each intervallo t interessato do
        verifica ammissibilità riga t dopo la complementazione;
    for each trial solution ts do
        if (ts feasible) e (supera tabu test o aspiration test) then
            /* nuovi c.o. = costi operativi */
            for each intervallo t interessato do
                if non ancora calcolato il nuovo c.o. then
                risolvi l'EDP;
                memorizza il c.o. calcolato;
                endif
            endfor
            cts = calcola costo totale di ts;
            if cts < cbfs then bfs = ts; cbfs = cts;
        endif
    endfor
endfor
```

Figura 0.4 - *Algoritmo per la determinazione della miglior soluzione ammissibile del neighbourhood della soluzione corrente.*

Fino a quando non si verificano le condizioni per terminare l'esecuzione, l'algoritmo ripete le seguenti operazioni:

1) incrementa le variabili preposte al conteggio delle iterazioni;
2) analizza il neighbourhood della soluzione corrente e determina la miglior soluzione ammissibile non vietata dalla tabu list (o eventualmente accettabile comunque in base al criterio di aspirazione). Tali scelte sono operate dall'oggetto coordinatore descritto nel precedente paragrafo;
3) se la soluzione ammissibile del neighbourhood così determinata ha costo totale inferiore alla miglior soluzione, viene aggiornata la miglior soluzione.

L'algoritmo termina quando la miglior soluzione individuata non cambia da un certo numero di iterazioni (ad esempio 200) o quando viene raggiunto il massimo numero di iterazioni possibili (ad esempio 1000). In figura 0.5 si riporta sinteticamente l'algoritmo.

```
inizializzazione;
determina una soluzione iniziale ammissibile con euristico;
best_sol = soluzione corrente;
couter = 0; do_not_improve = 0;
do
    counter ++; do_not_improve ++;
    new_sol =  migliore soluzione ammissibile appartenente al
               neighbourhood e non tabu;
    if costo(new_sol) < costo(best_sol) then
        do_not_improve = 0;
        best_sol = new_sol;
    endif
    new_sol è la nuova soluzione corrente;
while (counter > 1000) o (do_not_improve > 200);
```

Figura 0.5 - *Algoritmo TS per lo unit commitment.*

I.7.3 Descrizione delle TL proposte

Per completezza si è deciso di implementare due approcci che prevedono una TL per ogni centrale, mentre altri due approcci utilizzano una unica TL per tutti i generatori (si veda la tabella 0.3).

	TL	*Viene memorizzato:*	*Una mossa è TABU se:*
1	1	unità interessata dalla mossa	medesima unità
2	1	estremi della sequenza modificata e unità interessata	medesima unità e sequenza si sovrappone
3	1 ∀unità	numero di 1 nella colonna interessata dalla mossa	stesso numero di 1 dopo la mossa
4	1 ∀unità	estremi della sequenza modificata	ha medesimi estremi

Tabella 0.3 - *I differenti approcci proposti.*

I.8. Risultati sperimentali

L'algoritmo realizzato è stato testato con numerose istanze numeriche. Alcune di queste sono state rintracciate in letteratura e costituiscono un riferimento importante per qualsiasi lavoro inerente la generazione di energia. Altre sono state generate da un programma appositamente scritto, che, partendo da informazioni istanze note [3], genera nuove unità modificando opportunamente determinati coefficienti [5].

I.8.1 Prima istanza

Un algoritmo di simulated annealing [10] è stato testato sull'istanza di Bard [3]. I costi di start-up non sono esponenziali ma costanti, ed in particolare pari al coefficiente b_2 dell'istanza di riferimento. L'algoritmo è stato adattato per realizzare costi di accensione costanti, e quindi ottenere risultati sperimentali confrontabili direttamente.

```
0 1 5 4 0 4 0 4 5 0   prodcost:10551,1050
0 1 0 0 0 1 0 4 5 0    prodcost:9703,9997
0 1 0 0 0 1 0 0 1 0    prodcost:8796,8896
0 1 0 0 0 1 0 0 1 0    prodcost:8350,6538
0 1 0 0 0 1 0 0 1 0    prodcost:9934,4518
0 1 0 0 0 1 0 1 1 0   prodcost:13485,6705
0 1 0 1 0 1 0 1 1 0   prodcost:19127,2988
0 1 1 1 0 1 0 1 1 0   prodcost:23749,4705
0 1 1 1 0 1 0 1 1 1   prodcost:28253,9463
0 1 1 1 0 1 1 1 1 1   prodcost:31701,7354
0 1 1 1 0 1 1 1 1 1    prodcost:33219,8011
0 1 1 1 0 1 1 1 1 1   prodcost:34242,0440
0 1 1 1 0 1 1 1 1 1   prodcost:32965,5022
0 1 0 1 0 1 1 1 1 1   prodcost:29414,3534
0 1 0 1 0 1 1 1 1 1   prodcost:26865,7082
0 1 0 1 0 1 1 1 1 1   prodcost:25370,5138
0 1 0 1 0 1 1 1 1 1    prodcost:27117,3872
0 1 1 1 0 1 1 1 1 1   prodcost:32205,7133
0 1 1 1 0 1 1 1 1 1   prodcost:33219,8011
0 1 0 1 0 1 1 1 1 1   prodcost:28898,9613
0 1 0 1 0 1 0 1 1 1   prodcost:20698,4038
0 1 0 0 0 1 0 1 1 0   prodcost:15878,1731
0 1 0 0 0 1 0 1 1 0    prodcost:12572,8453
0 1 0 0 0 1 0 0 1 0   prodcost:11110,6932
Total production cost: 527435,1234
Total start-up cost  : 5425,
Total cost           : 532860,1234
```

Figura 0.6 – *Output dell'al-goritmo relativo alla soluzione ammissibile di minor costo individuata. Ogni colonna indica lo stato di una centrale, ogni riga indica quali centrali sono accese in un dato intervallo.*

La miglior soluzione individuata è riportata in figura 0.6. Si desidera evidenziare che i valori 4 e 5 nello schedule rappresentano lo **0** e l'**1** rispettivamente, ovvero quei valori comuni a tutte le soluzioni ammissibili e che quindi non possono mai essere modificati, valori forzati dai vincoli sullo stato iniziale. In figura 0.7 si riporta la potenza erogata da ogni centrale.

unit							
2	*3*	*4*	*6*	*7*	*8*	*9*	*10*
400	183,6359	0	0	0	0	441,3631	0
400	0	0	200	0	0	399,9999	0
400	0	0	185,6922	0	0	314,3084	0
386,6885	0	0	178,3886	0	0	284,9231	0
400	0	0	200	0	0	425,0001	0
400	0	0	200	0	375	425,0001	0
400	0	409,7483	200	0	375	585,2519	0
400	360,509	420	200	0	375	644,4908	0
400	468,0628	420	200	0	375	768,0096	218,9275
400	444,5936	420	200	358,0554	375	741,0567	211,2948
400	486,309	420	200	404,8659	375	788,9643	224,8615
400	514,1188	420	200	436,0722	375	820,902	233,9058
400	479,3562	420	200	397,0638	375	780,9794	222,6003
400	0	420	200	463,7907	375	849,2701	241,9393
400	0	420	200	355,7178	375	738,6644	210,6173
400	0	420	200	290,8748	375	672,3015	191,8243
400	0	420	200	366,5251	375	749,7249	213,7495
400	458,4988	420	200	373,6589	375	757,0259	215,817
400	486,309	420	200	404,8659	375	788,9643	224,8615
400	0	420	200	442,1761	375	827,149	235,6749
400	0	404,8862	200	0	375	579,5541	165,5595
400	0	0	200	0	375	675,0004	0
400	0	0	191,6371	0	370,1353	338,2276	0
400	0	0	200	0	0	549,9992	0

Figura 0.7 – *Le unità 1 e 5 sono spente sempre.*

L'algoritmo TS *migliora la soluzione precedentemente individuata*: il costo passa da 536622$ a 532860$. *Tale miglioramento è inoltre ottenuto rispettando un numero maggiore di vincoli*. Confrontando infatti le due soluzioni proposte si può notare che, a differenza di quella precedentemente proposta, la soluzione individuata

dall'algoritmo TS realizzato rispetta anche i vincoli sullo stato iniziale.

Un altro aspetto importante da sottolineare riguarda il tempo di esecuzione. Anche se non è possibile un confronto diretto, in quanto gli algoritmi andrebbero testati sulla medesima macchina, è possibile rilevare la grande differenza esistente tra il tempo di calcolo dell'algoritmo di SA precedentemente proposto (46.52 minuti) e quello dell'algoritmo TS realizzato (poco più di 2 secondi su un Pentium III Intel 450 MHz). La differenza è talmente rilevante che è possibile giungere alla conclusione che *l'algoritmo TS è molto più veloce* anche se le macchine non sono le medesime (si parla comunque di PC costruiti negli stessi anni).

Eseguendo l'algoritmo utilizzando diversi ordinamenti (i sei proposti), euristici (i 3 precedentemente descritti) e lunghezze della TL (da 10 a 30, TL del 2 tipo di tabella 0.3) si ottengono 21·3·6=378 soluzioni ammissibili finali. Aggregando tali soluzioni in base al costo totale (approssimandolo per difetto quest'ultimo al migliaio più vicino) si ottiene il grafico 0.1.

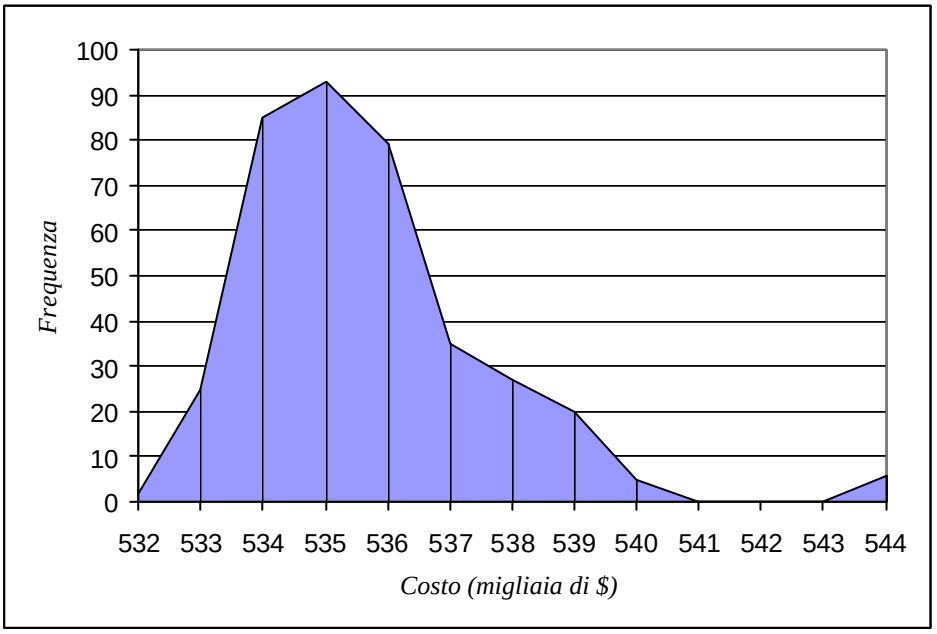


Grafico 0.1 – *Le 378 soluzioni sono state aggregate arrotondando per difetto al migliaio più vicino. In ordinata è riportato il numero di volte in cui la soluzione ha avuto il costo in ascissa.*

Dal grafico si evince che l'algoritmo TS *migliora la precedente soluzione molto spesso* (indipendentemente dall'ordinamento delle colonne, dall'euristico iniziale, dalla lunghezza della TL). Più precisamente, 248 volte su 378 (il 65,6%) l'algoritmo TS migliora la precedente soluzione.

Il grafico 0.2 mostra come l'*ottima local search* porta molto rapidamente ad un minimo locale, al quale corrisponde un costo già molto interessante. L'algoritmo poi "scappa" da questo minimo accettando soluzioni peggioranti. Il risultato è una oscillazione, ben evidente nel grafico, del costo della soluzione corrente. Il *buon funzionamento dell'algoritmo TS* è dimostrato dal fatto che non si generano loop (il tratto più fine in figura non diventa mai periodico).

Il confronto delle diverse TL (grafico 0.3) mostra come gli approcci 2 e 4 producano i migliori risultati. *Le soluzioni determinate sono comunque molto vicine in termini di costo* (il gap massimo è pari allo 0,17%), ulteriore dimostrazione dell'ottima local search.

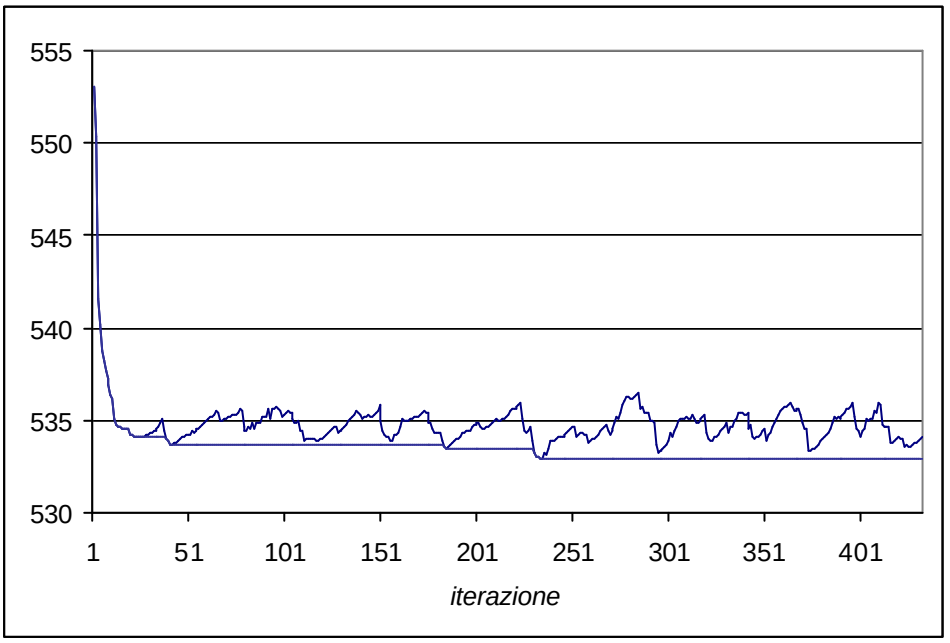


Grafico 0.2 – *Costo (in migliaia di $) della soluzione corrente, tratto più fine, e della miglior soluzione determinata.*

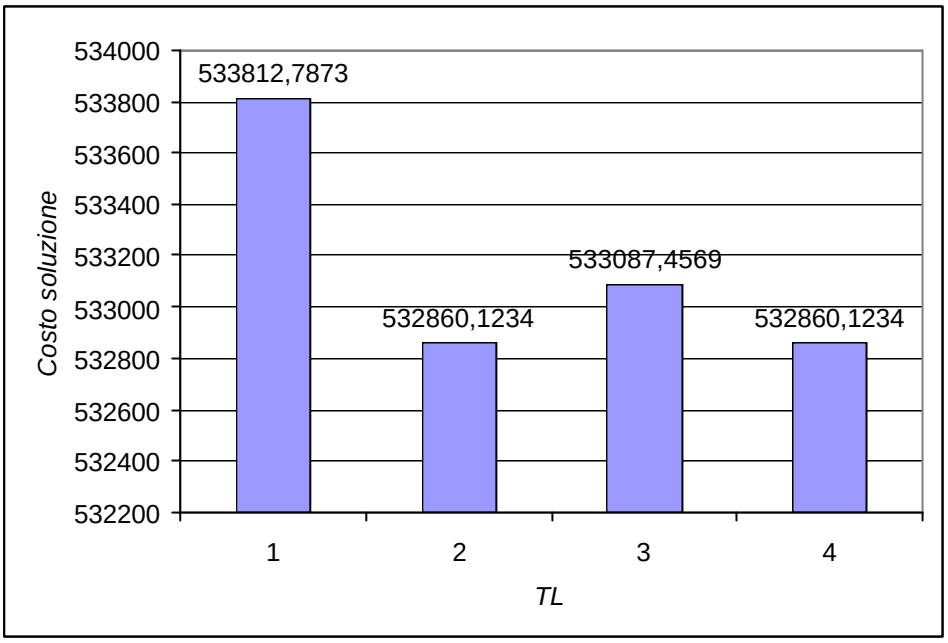


Grafico 0.3– *Confronto tra le quattro diverse TL. In ordinata costi in migliaia di $.*

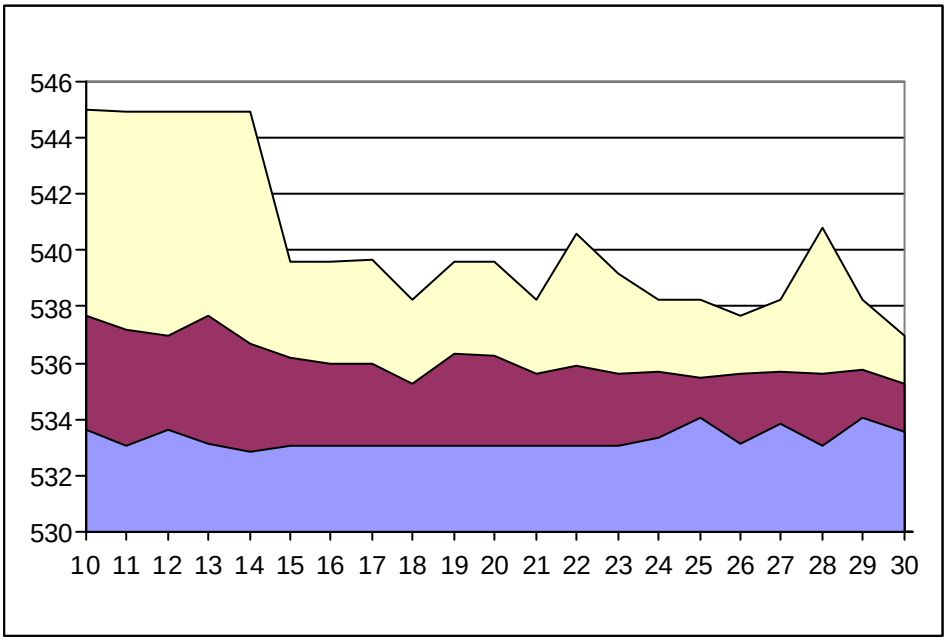


Grafico I.4 – *Per ogni valore della lunghezza della TL si riportano, dal basso verso l'alto, il costo della miglior soluzione, il costo medio, il costo della peggior soluzione (su 18 esecuzioni con diversa soluzione iniziale ottenute utilizzando diversi ordinamenti delle unità e diversi euristici). In ordinata costi in migliaia di $. Il miglior risultato si ottiene con z = 14.*

I.8.2 Seconda istanza

L'istanza [22] prevede 10 generatori e l'orizzonte temporale di interesse è costituito da 24 intervalli. Non vengono considerati vincoli sullo stato iniziale. La miglior soluzione individuata è riportata in figura I.8.

```
1 1 1 1 1 1 1 1 1 1   prodcost:3049,2224
1 1 1 1 1 0 1 1 1 1   prodcost:2856,3094
1 1 1 1 1 0 1 1 1 1   prodcost:2694,4304
1 1 1 1 1 0 1 1 1 1   prodcost:2664,1844
1 1 1 1 1 0 1 1 1 1   prodcost:2634,1485
1 1 1 1 1 0 1 1 1 1   prodcost:2727,0769
1 1 1 1 1 0 1 1 1 1   prodcost:2856,3094
1 1 1 1 1 0 1 1 1 1   prodcost:2727,0769
1 1 1 1 1 0 1 1 1 1   prodcost:2634,1485
1 1 1 1 1 0 1 0 1 1   prodcost:2572,3393
1 1 1 1 1 0 1 0 1 1   prodcost:2470,3657
1 1 1 1 1 0 1 0 1 1   prodcost:2437,0171
1 1 1 1 1 0 1 0 1 1   prodcost:2375,6308
1 1 1 1 1 0 1 0 1 1   prodcost:2340,9978
1 1 1 1 1 0 1 0 1 1   prodcost:2310,9168
1 1 1 1 1 0 1 0 1 1   prodcost:2249,2152
1 1 1 1 1 0 1 0 1 1   prodcost:2188,2267
1 1 1 1 1 0 1 0 1 1   prodcost:2127,9468
1 1 1 1 1 0 1 0 1 1   prodcost:2037,8399
1 1 1 1 1 0 1 0 1 1   prodcost:2007,5494
1 1 1 1 1 0 1 0 1 1   prodcost:1977,5234
1 1 1 1 1 0 1 0 1 1   prodcost:2097,0444
1 1 1 1 1 0 1 0 1 1   prodcost:2219,6853
1 1 1 1 1 0 1 0 1 1   prodcost:3111,7404
Total production cost: 59366,9469
Total start-up cost  : 0,
Total cost           : 59366,9469
```

Figura I.8 – *Output dell'al-goritmo relativo alla soluzione ammissibile di minor costo individuata. Ogni colonna indica lo stato di una centrale, ogni riga indica quali centrali sono accese in un dato intervallo.*

Indipendentemente dalla soluzione ammissibile iniziale (ottenuta dai diversi ordinamenti delle colonne e dall'euristico utilizzato) e dalla lunghezza della TL, *l'algoritmo TS individua sempre la miglior soluzione*. Nel grafico I.5 viene riportata la frequenza con la quale l'algoritmo determina le soluzioni di coso diverso. Nei quattro casi in cui la soluzione migliore risulta essere diversa, viene comunque ottenuto uno schedule dal costo molto vicino al migliore. L'istanza in esame si rivelata quindi particolarmente "semplice" per l'algoritmo TS realizzato.

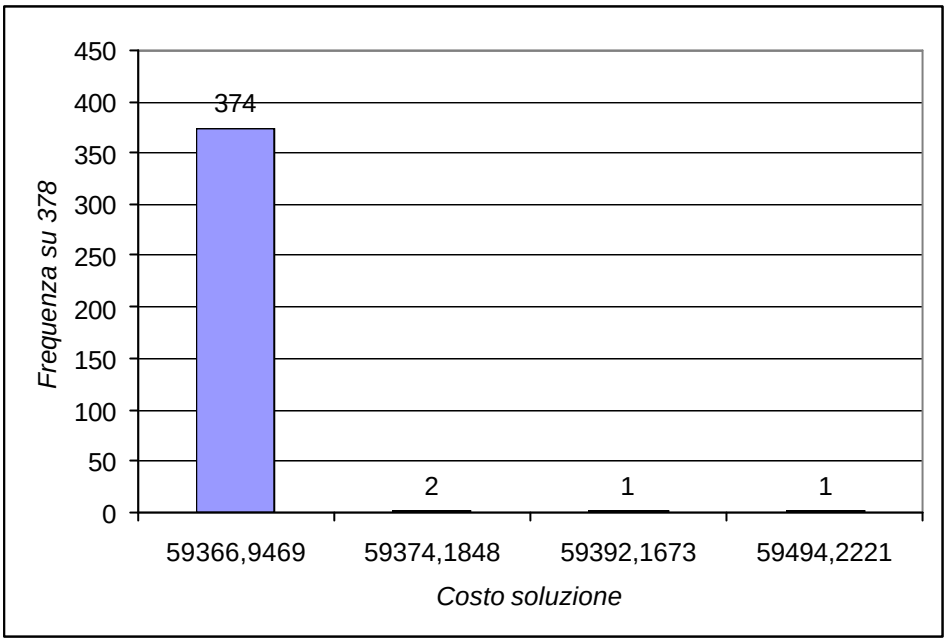


Grafico I.5 – *In ordinata è riportato il numero di volte in cui l'algoritmo determina le soluzioni di costo indicato in ascissa.*

	Costo totale *$*	CPU time *secondi*	Iterazioni
SSA [26]	59512	3162	
SSA [10]	59385	1812	
IP [22]	60667		
TSA [5]	59512		616
GA [22]	59385		
GTS [7]	59385		
TS	**59367**	**2**	**8**

Tabella I.4 - *Confronto con i risultati ottenuti da algoritmi presentati in altri lavori.*

Nella tabella I.4 si riporta in forma sintetica il confronto dei risultati ottenuti dall'algoritmo TS con altri algoritmi presentati in altrettanti lavori. *L'algoritmo proposto migliora i risultati di tutti i*

lavori che hanno utilizzato l'istanza in esame come metro di valutazione delle performance. Tale miglioramento è inoltre ottenuto con un minore sforzo computazionale. Valendo le considerazioni fatte nel precedente paragrafo, anche se non è possibile un confronto diretto è comunque possibile affermare che *l'algoritmo TS è molto più veloce*. Inoltre, confrontato i due algoritmi TS si nota che l'algoritmo proposto necessita di *pochissime iterazioni per convergere alla soluzione ammissibile finale*.

I.8.3 Terza istanza

Si considera ora l'istanza proposta in [3] con costi di start-up espressi correttamente, ovvero tramite funzione esponenziale. Il confronto viene eseguito tra l'algoritmo TS realizzato e tre algoritmi proposti in altri lavori. Per rendere i risultati omogenei si sono rilassati i vincoli sullo stato iniziale. Infatti in questi tre lavori le soluzioni proposte non risultano ammissibili se si considerano le limitazioni imposte dal vincolo in questione. Per una discussione più approfondita in merito a queste problematiche si rimanda al capitolo conclusivo. La miglior soluzione individuata è riportata in figura I.8.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
Total production cost: 528340,0259
Total start-up cost  : 12446,4166
Total cost           : 540786,4426
```

Figura I.8 – *Soluzione ammissibile di minor costo individuata. Ogni riga indica lo stato di una centrale (1 in alto, 10 in basso), ogni colonna indica quali centrali sono accese in un dato intervallo (primo intervallo a sinistra).*

Per completezza si è eseguito l'algoritmo senza rilassare i vincoli sullo stato iniziale. L'output è mostrato in figura I.9. I valori 4 e 5 nello schedule rappresentano lo **0** e l'**1** rispettivamente, ovvero quei valori comuni a tutte le soluzioni ammissibili e che quindi non possono mai essere modificati, valori forzati proprio dai vincoli sullo stato iniziale.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
4 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
4 4 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
Total production cost: 528956,6791
Total start-up cost  : 12667,3342
Total cost           : 541624,0133
```

Figura I.9 – *Soluzione ammissibile di minor costo individuata se non vengono rilassati i vincoli sullo stato iniziale. Ogni riga indica lo stato di una centrale (1 in alto, 10 in basso), ogni colonna indica quali centrali sono accese in un dato intervallo (primo intervallo a sinistra).*

Nella tabella I.4 si riporta in forma sintetica il confronto dei risultati ottenuti dall'algoritmo TS con altri algoritmi presentati in altrettanti lavori. *L'algoritmo proposto migliora i risultati* di tutti i lavori che hanno utilizzato l'istanza in esame come metro di valutazione delle performance. Tale miglioramento è inoltre ottenuto con un minore sforzo computazionale. Infatti *l'algoritmo TS è molto più veloce* (valgono le considerazioni dei precedenti paragrafi) e *necessita di meno iterazioni per convergere alla soluzione ammissibile finale*.

Nel grafico I.5 si riportano i risultati ottenuti nelle 252 esecuzioni (6 ordinamenti, 2 euristici, 21 lunghezze della TL 2). Le soluzioni sono aggregate grazie all'approssimazione per difetto al migliaio. Non è stato sfruttato il corretto dimensionamento di *z* possibi-

le dopo l'analisi delle precedenti istanze. Nel grafico I.6 si riportano i costi delle 12 soluzioni individuate ponendo $z = 14$ (dato precedentemente ricavato).

	Costi *($)*			CPU time *(secondi)*	Iterazioni
	totale	operativi	start-up		
SSA [10]	547708	532498	15210	2791	
TSA [3]	542442	528295	14146		924
GTS [7]	542931	528708	14222		
TS *no vincoli*	**540786**	**528340**	**12446**	**3**	**222**
TS *vincoli*	**541624**	**528957**	**12667**	**2**	**170**

Tabella I.4 - *Confronto con i risultati ottenuti da algoritmi presentati in altri lavori.Per completezza si riportano i risultati ottenuti sia rilassando per eliminazione i vincoli sullo stato iniziale sia considerandoli.*

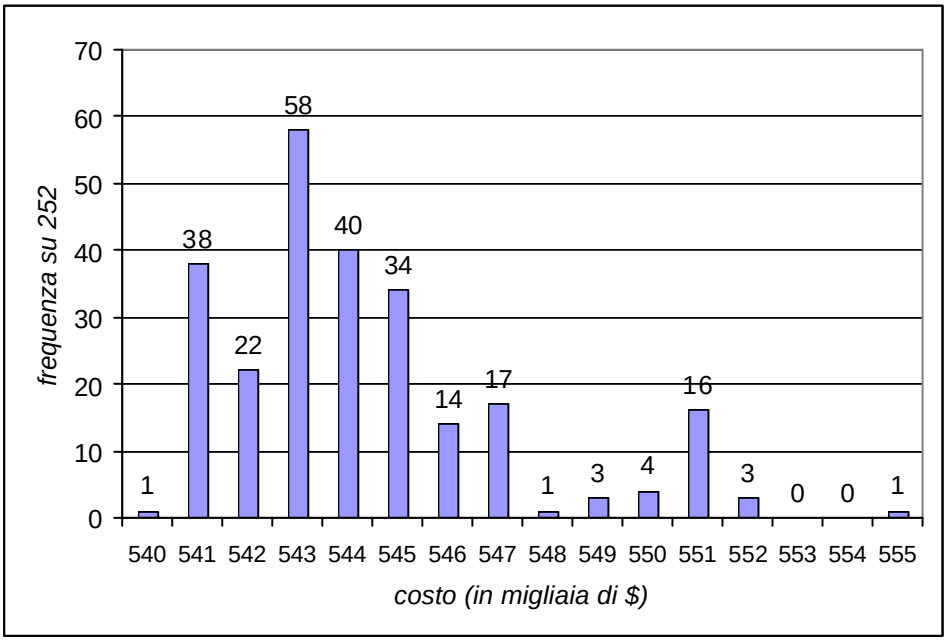


Grafico I.5 – *L'algoritmo è stato eseguito 252 volte con euristici, ordinamento delle unità e lunghezze TL diverse (quindi non ottimizzato).*

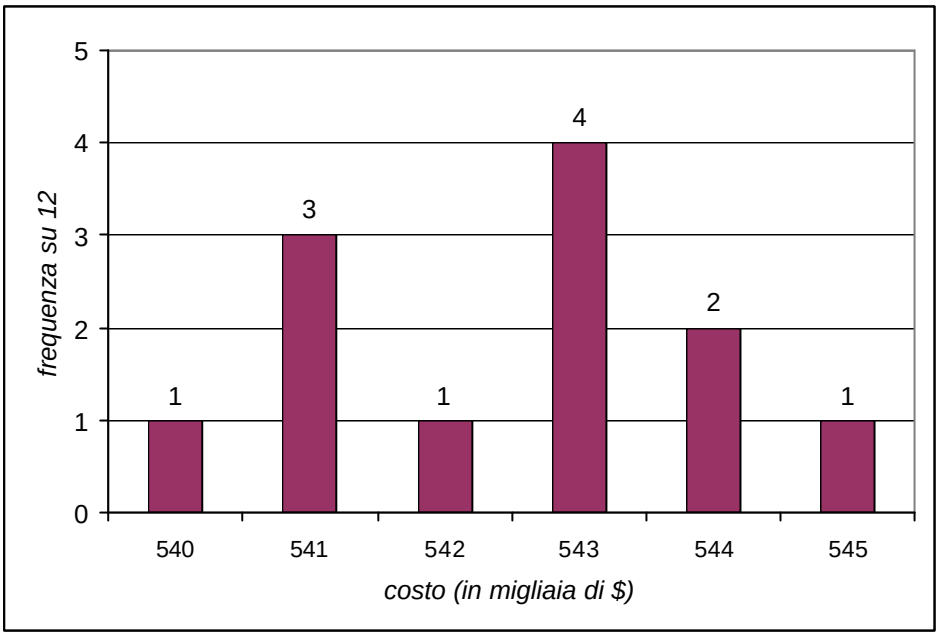


Grafico I.6 – *Sono riportati i 12 risultati ottenuti mantenendo la lunghezza della TL costante e uguale a 14.*

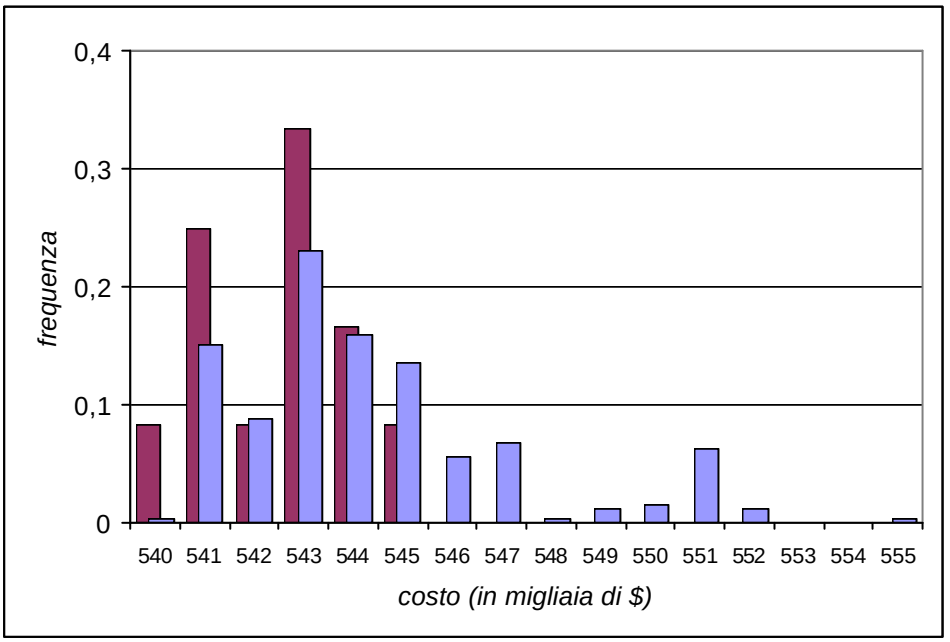


Grafico I.7 – *Le barre più chiare sintetizzano i risultati ottenuti con lunghezza della TL variabile, mentre quelle più scure quelli con z=14.*

Per ottenere dati comparabili sono state normalizzare le frequenze (dividendo quelle dei grafici I.5 e I.6 per 252 e 12 rispettivamente). Dal grafico I.7 si evince che un corretto dimensionamento della lunghezza della TL comporta uno spostamento dei risultati ottenuti verso costi minori. In altre parole, *un opportuno valore di z consente di selezionare esclusivamente le migliori soluzioni*. E' interessante osservare che le 12 esecuzioni con $z = 14$ individuano soluzioni ammissibili di costo distante non più dell'1% dalla miglior soluzione. *L'algoritmo quindi individua sempre buone soluzioni*.

I.8.4 Istanze di grandi dimensioni

Sono stati considerati tutti i vincoli del modello matematico proposto in questo lavoro, ovvero vincoli sullo stato iniziale, vincoli di minimo up/down time, vincoli sulla potenza erogabile da ogni unità, vincoli di load demand e spinning riserve, costi di start-up esponenziale e costi operativi quadratici. Si è cioè utilizzato il modello più preciso e con maggiori vincoli.

L'algoritmo proposto lavora sempre molto bene. Gli euristici hanno sempre individuato una soluzione iniziale ammissibile, indipendentemente dall'ordine delle unità. L'algoritmo TS ha poi sempre migliorato rilevantemente tale soluzione. All'aumentare delle dimensioni cresce ovviamente il tempo di calcolo (grafico I.8), a causa del neighbourhood di dimensioni maggiori.

Per istanze di 50 unità il tempo necessario è ancora molto inferiore al minuto, e può essere ulteriormente ridotto intervenendo sul criterio che porta all'arresto dell'algoritmo (ad esempio ridurre da 200 a 50 il numero di iterazioni possibili senza miglioramenti della best solution).

Relativamente ad una istanza di 50 unità, è stato eseguito l'algoritmo TS cambiando la lunghezza della TL, partendo sempre dalla medesima soluzione ammissibile iniziale. Dal grafico I.9 si evince che, per maggiori dimensioni del problema, si ottengono soluzioni migliori se si aumenta la lunghezza della TL. Tale risultato era prevedibile in quanto è stato calcolato sperimentalmente [11] che la dimensione della TL che produce buoni risultati spesso aumenta con

la dimensione del problema.

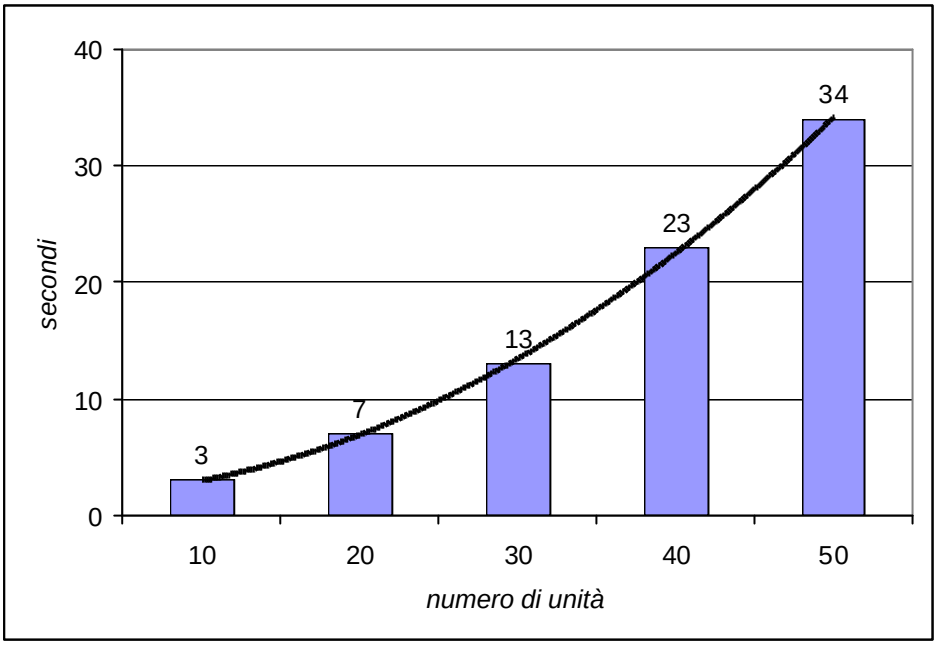


Grafico I.8 – *Tempo di calcolo richiesto in base alla dimensione del problema. Pc utilizzato: Pentium III Intel 450 MHz.*

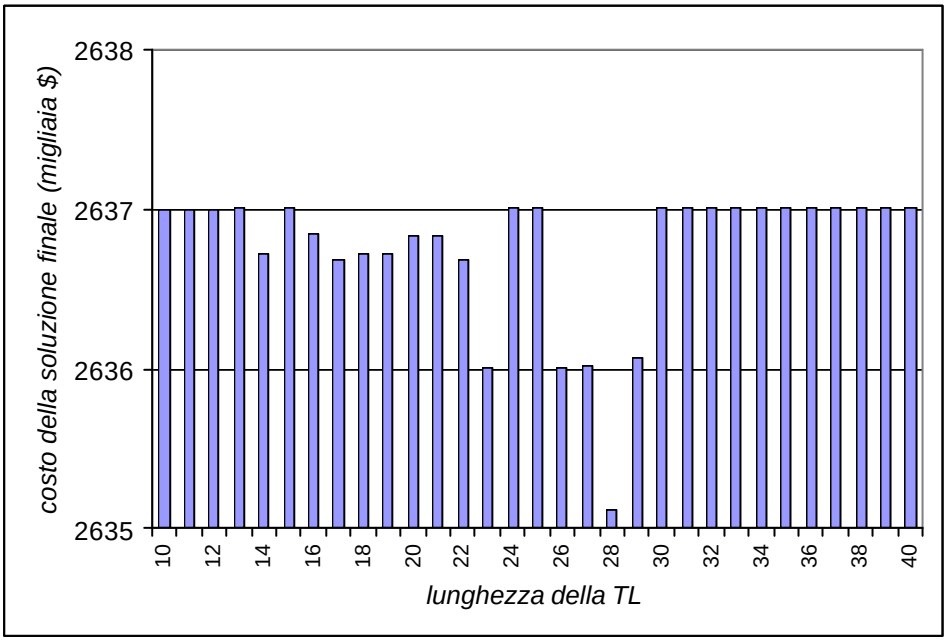


Grafico I.9 – *Costo della soluzione finale ottenuta per ogni lunghezza della TL, a partire dalla medesima soluzione ammissibile iniziale.*

Nel caso in esame la soluzione di costo inferiore viene ottenuta per $z = 28$. In ogni caso tutte le *soluzioni ottenute differiscono al più dello 0.076%.* In altre parole, il dimensionamento della TL incide in modo poco rilevante sul costo finale della soluzione. Tale risultato dimostra inequivocabilmente l'ottimo funzionamento dell'algoritmo realizzato.

Dimensionando la TL a 28 (valore che porta il risultato migliore) si è poi eseguito l'algoritmo TS diversificando 17 volte, ovvero eseguendolo 18 volte e cominciando la computazione da soluzioni iniziali ammissibili diverse, posizionate cioè in punti distinti nello spazio di ricerca. Le soluzioni iniziali sono state ottenute dai diversi ordinamenti e euristici. Nel grafico I.10 e in tabella I.5 sono riportati i risultati ottenuti. *Tutte le soluzioni differiscono pertanto al più per lo 0,59%.*

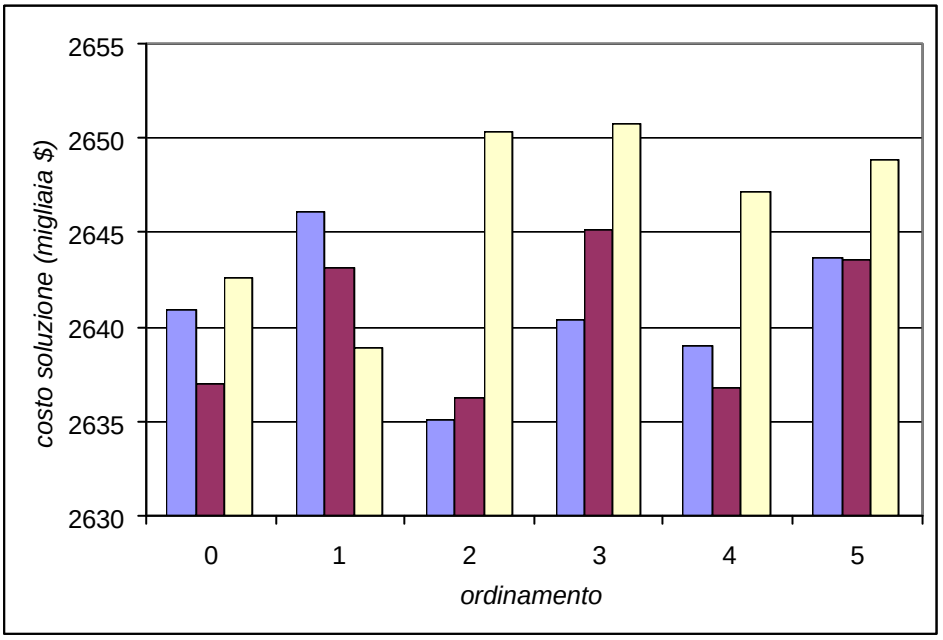


Grafico I.10 - *Costo della soluzione finale ottenuta a partire da diverse soluzioni iniziali. In ascissa sono riportati i 6 diversi ordinamenti; a colonne di colore diverso corrispondono euristici diversi. La miglior soluzione corrisponde a ordinamento per rapporti crescenti di potenza massima e costo operativo associato ed euristico 1 descritto nel paragrafo 6.3.*

		Euristico		
		1	2	3
Ordinamento	0	2640929	2637072	2642598
	1	2646072	2643137	2638875
	2	2635120	2636275	2650345
	3	2640457	2645205	2650788
	4	2638962	2636787	2647157
	5	2643648	2643525	2648785

Tabella I.5 - *Costo ($) della soluzione finale ottenuta a partire da diverse soluzioni iniziali, ottenute ricorrendo a diversi euristici e ordinamenti delle unità.*

I.9. I principali contributi in sintesi

- Analizzati ed implementati 2 algoritmi per l'EDP:
  - migliorati entrambi (§ 5.2.1, 5.2.3);
  - si è verificato che uno non funziona correttamente in determinati casi (§ 5.2.1);
  - integrazione ottimale nell'algoritmo TS (§ 5.4);
  - minimizzato il numero di invocazioni (§ 8.2.4, 8.4).
- Formalizzato il modello matematico sintetizzando il contributo di diversi lavori (§ 4.1-8).
- Problemi emersi dall'analisi dei casi proposti in letteratura (§ 10.3):
  - emersa una non corretta gestione dello stato iniziale (§ 10.3.1):
    - vincoli non rispettati;
    - istanze particolari che rilassano implicitamente tali vincoli;

    - inconsistenza di alcuni risultati (§ 10.3.2).
- Si sono proposti 3 euristici diversi per determinare una soluzione ammissibile (§ 6.3-5) e diversi ordinamenti delle unità (§ 6.7) per poter individuare molteplici soluzioni iniziali per l'algoritmo TS. I vantaggi di tale approccio sono legati a (§ 6.6):
    - ammissibilità (per ogni istanza e per qualsiasi ordinamento dei generatori gli euristici hanno sempre individuato una soluzione ammissibile);
    - diversificazione.
- Proposto un nuovo approccio (da risultato voluto → ammissibilità a ammissibilità → risultato voluto) (§ 7.1).
- Generatore di mosse ammissibili (§ 7.3):
    - migliora la computazione velocizzando la valutazione dell'ammissibilità;
    - l'ammissibilità non è più un risultato da ottenere ma un dato acquisito;
    - generazione esaustiva di tutte le mosse possibili, quindi analisi molto approfondita del neighbourhood.
- Caratteristiche dell'algoritmo:
    - elasticità:
        - possibilità di implementare costi di start-up costanti ed esponenziali (con diverse formulazioni);
        - possibilità di rilassare a piacimento e senza alcuna difficoltà alcuni vincoli (sullo stato iniziale, sulla riserva di potenza, eliminare costi di start-up).
    - espansibilità, ovvero possibilità di aggiungere ulteriori vincoli modificando in minima parte l'algoritmo (vincoli di rampa sono immediati).
- Proposti 4 diversi approcci per le TL secondo due filosofie principali (una TL unica o una per ogni unità) (§ 8.6).
- Proposta una tecnica di calcolo dei costi "incrementale" (§ 8.3) e i conseguenti accorgimenti (§ 8.3.1) per gestire correttamente la conseguente perdita di informazione (rumore e refresh).

- Ottimi risultati sperimentali (§ 9.1-4):
    - sia per problemi di piccole dimensioni che per problemi di grandi dimensioni;
    - migliorate le soluzioni precedentemente individuate;
    - minor tempo di elaborazione;
    - minor numero di iterazioni per determinare la soluzione finale;
    - ottima local search;
    - la scelta progettuale relativa alla lunghezza della TL non è più cruciale per la qualità della soluzione finale;
    - l'algoritmo non deve essere "fortunato" per determinare una buona soluzione finale.

I.10. Bibliografia

[1] Hadi Saadat, "Power system analysis.",McGraw-Hill, 1999, pp. 266-279.

[2] Mantawy, A.H., Abdel-Magid, Y.L., Selim, S.Z., "Unit commitment by tabu search", *IEE Proc.-Gener. Transm. Distrib.*, Vol. 145, No. 1, January 1998, pp. 56-64.

[3] Jonathan F. Bard, "Short-term scheduling of thermal-electric generators using lagrangian relaxation", *Operations Research*, Vol. 36, No. 5, September-october 1988, pp. 756-766.

[4] Hiroyuki Mori, Osamu Matsuzaki, "A parallel tabu search approach to unit commitment in power systems", *Proc. of 1999 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE SMC '99, Vol. 6, 1999, pp. 509-514.

[5] Bakirtzis, A.G., Zoumas, C.E., "Lambda of Lagrangian relaxation to unit commitment problem", *IEEE Proc.-Gener. Transm. Distribut.*, Vol. 147, No. 2, March 2000, pp. 131-137.

[6] Hiroyuki Mori, Takayuki Usami, "Unit commitment using tabu search with restricted neighborhood", *Proc. of ISAP '96*, No. 0221, Orlando, Florida, Usa, January 1996, pp. 422-428.

[7] Mantawy, A.H., Abdel-Magid, Youssef L., Selim, Shokri Z., “Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem”, *IEEE Transaction on Power System*, Vol. 14, No. 3, August 1999, pp. 829-836.

[8] Subir Sen, D.P. Kothari, “Optimal thermal generaing unit commitment: a review”, *Electrical Power & Energy Systems*, Vol. 20, No. 7, 1998, pp. 443-451

[9] Lauer, G.S., Bertsekas, D.P., Sandell, N.R., Posbergh, T.A., “Solution of large-scale optimal unit commitment problems”, *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, No.1 january 1982, pp. 79-81.

[10] Mantawy, A.H., Abdel-Magid, Y.L., Selim, S. Z., “A simulated annealing algorithm for unit commitment”, *IEEE Transaction on Power Systems*, Vol. 13, No. 1, February 1998, pp. 197-204.

[11] Bland, J.A., Dawson, G.P., “Tabu search and design optimization”, Comput. Aided Des., 1991, 23, (3), pp. 195-201.

[12] Allen J. Wood and Bruce F. Wollenberg, "Power generation operation and control", John Wiley & Sons, 1996.

[13] Silvano Martello, “Lezioni di Ricerca Operativa”, Progetto Leonardo, Bologna, 1995.

[14] Martello, S., “Soluzione approssimata di problemi di Ottimizzazione Combinatoria”, dottorato di ricerca, Bologna, giugno 2000.

[15] Bertsekas, D.P., Lauer, G.S., Sandell, N.R., Posbergh, T.A., ”Optimal Short-Term Scheduling of Large-Scale Power Systems“, *IEEE Transactions on Automatic Control*, Vol. AC-28, No. 1, January 1983, pp. 1-11.

[16] Berger, D., Gendron, B., Potvin, J.Y., Raghavan, S., Soriano, P., “Tabu search for a Network Loading Problem with Multiple Facilities”, *Journal of Heuristics*, 6 (2000), pp. 253-267.

[17] Cohen, A.I., Brandwajn, V., Chang, S.K., “Security Constrained Unit Commitment for Open Markets“, *Proceedings of the 21st 1999 IEEE International Conference on Power Industry Computer Applications*, PICA '99, pp.39-44.

[18] Lauer, G.S., Bertsekas, D.P., Sandell, N.R., Posbergh, T.A.,

"Solution of large-scale optimal unit commitment problems", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 1, January 1982, pp.79-86.

[19] Glover, F., Kochenberger, G., Alidaee, B., Amini, M., "Tabu search with critical event memory: an enhanced application for binary quadratic programs", *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman & C. Roucairol, Eds., Kluwer Academic Publishers, 1998, pp. 93-109.

[20] Allen J. Wood and Bruce F. Wollenberg, "Power generation operation and control", John Wiley & Sons, 1996.

[21] Dipankar Dagupta, "Unit commitment in thermal power generation using genetic algotithms", *Proceeding of Sixth International Conference & Ingeneering Applications of Artificial Intelligence and Expert Systems*, june, scotland, 1993.

[22] Turgeon, A., "Optimal Scheduling of Thermal Generating Units", *IEEE Trans. On Automatic Control*, Vol. AC-23, No. 6, 1978, pp. 1000-1005.

[23] Tong, S.K., Shahidehpour, S.M., Ouyang, Z., "A heuristic short-term unit commitment", *IEEE Transactions on Power Systems*, 1991, Vol. PWRS-6, No. 3, pp. 1210-1216.

[24] Bland, J.A., Dawson, G.P., "Tabu search and design optimization", Comput. Aided Des., 1991, 23, (3), pp. 195-201

[25] Mantawy, A.H., Abdel-Magid, Y.L., Selim, S. Z., "A genetic algorithm with local search for unit commitment", *Proceeding of the intelligent System Applications to Power Systems (ISAP '97)*, July 6-7, 1997, Korea, pp. 170-175.

[26] Zhuang, F., Galiana, F.D., "Unit commitment by Simulated annealing", *IEEE Trans. on Power Systems*, Vol. 5, No. 1, 1990, pp. 311-318.

I.10.1 Per ulteriori approfondimenti

[27] Glover, F., "A user's guide to tabu search", Ann. Oper. Res., 1993, Vol. 41, pp.3-28.

[28] Glover, F., Greenberg, H.J., “New approach for heuristic search: a bilateral linkage with artificial intelligence”, Eur. J. Oper. Res., 1989, Vol. 39, pp.119-130.
[29] Glover, F., “Future paths for integer programming and links to artificial intelligence”, Comput. Oper. Res., 1986, Vol. 13, No. 5, pp. 533-549.
[30] Glover, F., “Tabu search-part I”, ORSA J. Comput., 1989, Vol. 1, No. 3, pp. 190-206.
[31] Glover, F., “Tabu search-part II”, ORSA J. Comput., 1990, Vol. 2, No. 1, pp. 4-32.
[32] Glover, F., Laguna, M., “Tabu Search”, Kluwer Academic Publishers, Boston, MA, 1997.
[33] Glover, F., “Artificial intelligence, heuristic frameworks and tabu search”, Managerial Decision Econ., 1990, **11**, pp.365-375.