Università degli Studi di Cagliari

Corso di Laurea Magistrale in Informatica

Insegnamento: Decision Science

Unit Commitment problem

Professori:

Massimo Di Francesco

Francesco Gorgone

Studente:

Jacopo Mereu

matr. 60/73/65264

Anno Accademico 22/23

Cagliari - 24 agosto 2022

Sommario

Questa tesina racconta in maniera molta succinta un argomento molto trattato nel mondo della ricerca ed estremamente importante nel mondo dell'industria energetica: lo Unit Commitment Problem.

Nel 1° capitolo verranno presentata la nomenclatura utile per il resto della lettura.

Nel 2° si introdurrà un sottoinsieme di quello che è il problema nella realtà e si vedrà come formulare un modello matematico descrittivo.

Nell'ultimo capitolo verrà mostrato come implementare il modello su IBM CPLEX Optimization Studio, utilizzando dei dati presi da [1]. Il codice può essere scaricato dalla seguente repository github.

Indice

Capitolo 1

Nomenclatura

Acronimi

- UCP - Unit Commitment Problem
- EDP - Economic Dispatch Problem
- SUc - Start-up costs
- SDc - Shut-down costs
- f.o. - funzione obiettivo

Sinonimi

- Carico eneregetico/Domanda/Richiesta/Demand/Load
- Unità/Generatore
- Costi di avvio/start-up costs
- Costi di spegnimento/shut-down costs

Notazione matematica

- Variabili
  - u_{it} - stato dell'unità i al tempo t (1=up / 0=down)

  - p_{it} - potenza in output dell'unità i al tempo t

- Dati

  - I - numero di unità
  - T - numero di periodi temporali

  - $D(t)$ - domanda nel periodo t

  - P_i^{min} - soglia di output minima dell'unità i, se attiva
  - P_i^{max} - soglia di output massima dell'unità i, se attiva

  - $k_1(i)$ [1] - coefficiente di *no-load costs* fisso in $ dell'unità i
  - $k_2(i)$ [1] - coefficiente di *marginal costs* in [$/MWh] dell'unità i
  - $k_3(i)$ [1] - coefficiente di *marginal costs* in [$/MW^{2}h] dell'unità i

  - τ_i^u - numero minimo di stati *up* consecutivi in cui l'unità i deve rimanere up dopo l'accensione
  - τ_i^d - numero minimo di stati *down* consecutivi in cui l'unità i deve rimanere down dopo lo spegnimento

  - h_i - stato iniziale dell'unità

- Funzioni

  - $c_i(u_{it}, p_{it})$ - costo di produzione dell'unità i al tempo t
  - $su_i(u_{i(t-1)}, u_{it})$ - SUc dell'unità i al tempo t
  - $sd_i(u_{ir}, u_{it})$ con $r = max\{r' < t : u_{it} = 0\}$ - SDc dell'unità i al tempo t

[1] Nel paper originale si chiamano $c_1(i), c_2(i), c_3(i)$ ma ho pensato che potesse creare confusone con la funzione di costo di produzione c_i con $i = 1, 2, 3$.

Capitolo 2

Teoria del problema

Nello UCP viene richiesto di soddisfare un fabbisogno di energia elettrica, sfruttando *al meglio* i generatori che si hanno a disposizione per ridurre il più possibile i costi di produzione, sapendo che esistono una serie di vincoli che si devono rispettare. Le domande a cui dobbiamo trovare la risposta sono le seguenti:

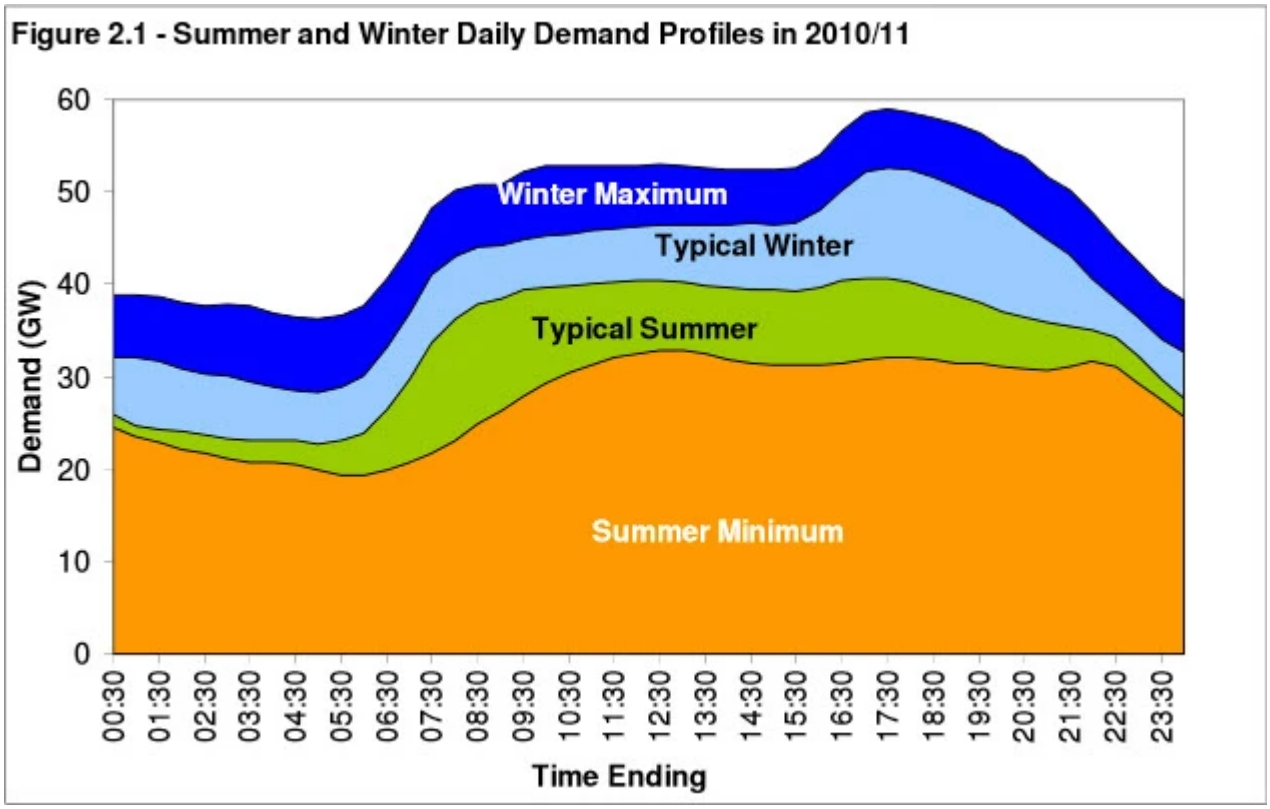
1. Quanto dovranno produrre le unità attive (sapendo che maggiore è l'output, maggiore è il costo)?

2. Quali unità dobbiamo attivare ma soprattutto quando? Questa è la differenza principale fra uno UCP e un EDP, in quanto nel secondo ci vengono assegnate le unità attive e dobbiamo solamente decidere quanto dovranno produrre.

2.1 La domanda

Se pensiamo alla quotidianità di una qualsiasi communità, è intuitivo pensare che ci sono degli orari in cui il consumo elettrico è basso (ad esempio dalle 01:00 alle 05:00) e altri momenti in cui il consumo elettrico è molto alto (le 08:00, le 13:00 o le 20:00).

Figura 2.1: Domanda suddivisa in 24 intervalli di 1 ora ciascuno



Per questo viene creato un programma suddiviso in intervalli, ad esempio da 24 o 48 ore (e.g. [00:00-00:59], [01:00-01:59], ...); la domanda conseguentemente diventa dipendente dal tempo ma per semplicità si assume costante all'interno degli intervalli (sia P =[00:00-00:59] un intervallo, $D(x) = D(x') \quad \forall x, x' \in P$).

2.2 Le unità

Si possono distinguere diversi tipi di unità a seconda della fonte di energia utilizzata (termica, nucleare, rinnovabile) e vien da sé che la tipologia dell'unità ha dei vincoli intrisici (vincoli sul flusso di acqua rilasciata nel canale di restituzione nel caso di centrali idroelettriche o sulla velocità rotazionale del rotore nel caso di pale eoliche).
In questa tesina mi concentrerò sul tipo di unità su cui mi sono "specializzato", ovvero quella termoelettrica, detta anche *flessibile* perché l'output delle unità di questo tipo è molto più gestibile (basta dosare il combustibile) di quello di una centrale nucleare (che per questioni di

sicurezza non può cambiare spesso la potenza in output) o di uno stabilimento di pale eoliche/pannelli solari (che dipendono da fattori naturali non controllabili dall'uomo).

Lo stato $u[i][t]$ in cui si può trovare un'unità i al tempo t sono due:

1. *Up*, ovvero l'unità è accesa e sta producendo.
2. *Down*, ovvero l'unità è spenta.

Per semplicità, nel modello, non viene considerato il tempo necessario per passare da uno stato all'altro (perché nella realtà ci vuole del tempo per passare da down ad up e viceversa), ma solamente i costi che queste transizioni comportano.

2.3 La funzione obiettivo

Un UCP ci chiede di minimizzare i costi del carburante utilizzato per far lavorare le unità, ovvero ci chiede di minimizzare una funzione che deve considerare i seguenti costi:

- I *no-load costs*, ovvero il costo del carburante che serve per mantenere l'unità attiva senza però produrre nulla. E' un costo fisso.
- I *marginal costs*, ovvero il costo del carburante che serve all'unità per produrre: maggiore è l'output dell'unità, maggiore sarà il carburante utilizzato e quindi maggiore sarà il costo di produzione.
- Un'unità termoelettrica ha bisogno di raggiungere una certa temperatura prima di poter essere pronta ad operare, ovvero per passare dallo stato down allo stato up: questi sono i SUc. Teoricamente sono valori che dipendono da quanto tempo è passata dall'ultima combustione, però per semplicità li ho visti spesso rappresentanti come costi fissi.
- Un'unità termoelettrica quando smette di produrre ha bisogno di scendere gradualmente ad una certa temperatura prima di poter essere spenta, ovvero per passare dallo stato up allo stato down: questi sono i SDc. Teoricamente sono valori che dipendono da quanto tempo è rimasta operativa l'unità, però negli esempi visti li ho sempre visti nulli.

La funzione da minimizzare diventa dunque:

$$c_i(u_{it}, p_{it}) + su_i(u_{i(t-1)}, u_{it}) + sd_i(u_{ir}, u_{it})$$
$$\text{con } c_i(u_{it}, p_{it}) = no\_load\_costs + (marginal\_costs \times output)$$

Nota: Esiste un *tradeoff* fra le unità grandi e quelle piccole: le prime tendono ad avere SUc e SDc maggiori rispetto alle seconde, in quanto hanno bisogno di più carburante per raggiungere le temperature desiderate, ma tendono ad avere costi marginali molto più bassi perché sono più efficienti. Per questo motivo si cerca di tenere attive il più possibile le unità grandi (addirittura si può pensare di farle lavorare poco la notte piuttosto che spegnerle completamente), mentre quelle piccole vengono attivate/spente spesso ma vengono tenute attive il meno possibile (solitamente sono utilizzate nei periodi di picco). Questa particolarità si potrà poi osservare anche nella soluzione del problema affrontato.

2.4 I vincoli

Adesso verranno presentati dei vincoli tipici di uno UCP. Quelli evidenziati in giallo sono presenti anche nel problema pratico affrontato. Introduco prima i vincoli che coinvolgono più unità, detti anche **System Constraints**.

V.1 **Demand satisfaction** Il vincolo più importante che si deve rispettare è il soddisfacimento della domanda ad ogni intervallo, ovvero la produzione delle I unità in un periodo t deve essere sufficiente da eguagliare il carico energetico richiesto, ma non superiore perché sarebbe uno spreco.

$$\sum_{i=1}^{I} p_{it} = D(t) \quad \forall t \in T$$

V.2 **Reserve generation capacity** E' naturale pensare che si possano verificare delle perdite nella produzione, ad esempio le unità dovranno subire della manuntenzione di tanto in tanto e non saranno disponibili. In questo caso si dovrebbe obbligare le altre unità a lavorare di più per sopperire l'output dell'unità mancante, ma questo è possibile solamente se non stanno già lavorando al massimo. Per risolvere questo problema si scelgono le unità in modo tale che, lavorando al massimo, superino la domanda almeno di un fattore

R, così facendo "si spera" che nel caso in cui venga a mancare un'unità, si riesca lo stesso a soddisfare la domanda.

$$\sum_{i=1}^{I} u_{it} P_i^{max} \geq D(t) + R(t) \quad \forall t \in T$$

Sorge dunque la domanda: come scelgo R? Se è troppo grande, avrò una soluzione molto costosa, se è troppo piccolo, verrebbe a meno la sua utilità. Una possibilità è di usare metodi deterministici, come assegnargli un valore pari al lavoro prodotto dall'unità più potente oppure una percentuale della domanda nel periodo di picco; in alternativa negli ultimi 20 anni sono stati studiati anche dei metodi probabilistici, come stimare la probabilità che un'unità smetta di operare: maggiore è la probabilità, maggiore sarà R.

V.3 Emission limits Per questioni ambientali la legge vieta agli stabilimenti di superare le emissioni di certe sostanze (e.g. ossido e biossido di azoto generati dai processi di combustione) in un periodo temporale.

V.4 Crew limits L'attivazione di un'unità deve essere seguita da del personale, perciò il numero di unità attivabili in un istante temporale dipende dal personale disponibile.

$$N\_workers\_for\_unit \times [\sum_{i=1}^{I} (u_{it} > u_{i(t-1)})] \leq Crew_t \quad \forall t \in T$$

Ora verranno presentati i vincoli che influenzano singole unità, ovvero gli **Unit Constraints**.

V.5 Maximum generating capacity Un'unità ha dei limiti fisici che le impediscono di produrre oltre ad una certa soglia.

$$p_{it} \leq u_{it} P_i^{max} \quad \forall t \in T, i \in I$$

V.6 Minimum generating capacity Un'unità, per poter lavorare in maniera stabile e sicura, deve poter produrre al di sopra di una certa soglia.

$$u_{it} P_i^{min} \leq p_{it} \quad \forall t \in T, i \in I$$

V.7 Maximum Up Rate Non si dovrebbe aumentare la produzione in maniera brusca in un breve periodo temporale per non danneggiare l'unità, solitamente questa soglia è espressa in percentuale della potenza massima dell'unità.

$$p_{it} - p_{i(t-1)} \leq \Delta P_i^{max} \quad \forall t \in T, i \in I$$

V.8 Maximum Down Rate Non si dovrebbe diminuire la produzione in maniera brusca in un breve periodo temporale per non danneggiare l'unità, solitamente questo soglia è espressa in percentuale della potenza minima dell'unità.

$$p_{i(t-1)} - p_{it} \leq \Delta P_i^{min} \quad \forall t \in T, i \in I$$

V.9 Minimum Up Time Un'unità, dopo essere stata attivata, non può essere subito spenta o ridurrebbe la sua durata vitale: bisogna aspettare τ_i^u intervalli (considerando anche il periodo in cui è stata attivata). Questo si traduce nel dire che, se vogliamo avere una variabile $u[i][t] = 1$, bisogna accertarsi che nell'intorno r di dimensione $\tau_i^u - 1$ non vi siano $u[i][r] = 0$.

$$u_{it} \geq (u_{ir} - u_{i(r-1)}) \quad \forall t \in T, r \in [t - \tau_i^u + 1, t - 1], i \in I$$

V.10 Minimum Down Time Un'unità, dopo essere stata disattivata, non può essere subito riattivata o ridurrebbe la sua durata vitale: bisogna aspettare τ_i^d intervalli (periodo di disattivazione incluso). Questo si traduce nel dire che, se vogliamo avere una variabile $u[i][t] = 0$, bisogna accertarsi che nell'intorno r di dimensione $\tau_i^d - 1$ non vi siano $u[i][r] = 1$.

$$u_{it} \leq (1 - u_{i(r-1)} + u_{ir}) \quad \forall t \in T, r \in [t - \tau_i^d + 1, t - 1], i \in I$$

V.11 Initial status. In quale stato si trovano le unità all'inizio del problema? Questo viene specificato tramite l'initial status *h*. Il segno $sign(h)$ indica lo stato (positivo = up, negativo = down), mentre il valore assoluto $abs(h)$ ci indica da quanti periodi si trova in quello stato.

Sia chiaro, *h* ovviamente non è un vincolo, ma allora perché l'ho inserito nei vincoli? La sua presenza porta a delle modifiche del problema: va a influenzare rispettivamente i SUc e SDc (quindi la f.o.), poiché se $u[i][1] = 1$ ma $h_i > 0$ allora non dobbiamo considerare i costi SUc_i, idem per i SDc_i se $u[i][1] = 0$ ma $h_i < 0$. Lo stato iniziale inoltre va a influenzare i vincoli **V.9** e **V.10** perché ci obbliga a considerare anche i periodi precedenti a quello iniziale.

Capitolo 3

Problema affrontato

Il problema affrontato è stato trovato nel materiale assegnatomi [1] che a sua volta era stato preso da [2], nello specifico ho utilizzato i dati del "Case 1": 10 unità, 24 periodi, 240 variabili booleane $u[i][t]$ e 240 variabili intere $p[i][t]$.
I dati sono i seguenti:

Figura 3.1: Dati delle I unità: $P_i^{max}, P_i^{min}, k_1(i), k_2(i), k_3(i), \tau_i^u, \tau_i^d, su_i(u_{i(t-1)}, u_{it}), h_i$

Unit	Pmax (MW)	Pmin (MW)	c_1 ($)	c_2 ($/MWh)	c_3 ($/MW^2-h)	min up time (h)	min down time (h)	start -up cost ($)	initial status (h)
1	455	150	1000	16.19	0.00048	8	8	9000	8
2	455	150	970	17.26	0.00031	8	8	10000	8
3	130	20	700	16.60	0.00200	5	5	1100	-5
4	130	20	680	16.50	0.00211	5	5	1120	-5
5	162	25	450	19.70	0.00398	6	6	1800	-6
6	80	20	370	22.26	0.00712	3	3	340	-3
7	85	25	480	27.74	0.00079	3	3	520	-3
8	55	55	660	25.92	0.00413	1	1	60	-1
9	55	55	665	27.27	0.00222	1	1	60	-1
10	55	55	670	27.79	0.00173	1	1	60	-1

Figura 3.2: Dati della domanda $D(t)$

Hour	demand (MW)
1	800
2	850
3	950
4	1050
5	1100
6	1200
7	1250
8	1300
9	1400
10	1500
11	1550
12	1600

Hour	demand (MW)
13	1500
14	1400
15	1300
16	1150
17	1100
18	1200
19	1300
20	1500
21	1400
22	1200
23	1000
24	900

La f.o. è come quella formulata nel capitolo precedente, in cui:

- La funzione dei costi di produzione è quadratica (di conseguenza anche la f.o. sarà quadratica) perché assume la forma $c_i(u_{it}, p_{it}) = k_1(i)p_{it}^2 + k_2(i)p_{it} + k_3(i)$
- Gli SUc sono fissi.
- I SDc sono nulli.

I vincoli richiesti dal problema sono quelli sopra trattati **V.1**, **V.5**, **V.6**, **V.9**, **V.10**, **V.11**.

Il problema trattato è un MIQP ed è stato modellato 2 volte su CPLEX Optimization Studio usando il linguaggio Optimization Programming Language. Perché 2 volte? Perché il 1° modello era erroneamente non eseguibile tramite CPLEX e dovetti eseguirlo con CP, poi ho scoperto che potevo, con una semplice modifica, eseguirlo sfruttando CPLEX.

3.1 CP vs CPLEX

CPLEX Optimization Studio ci mette a disposizione 2 engines che permettono di risolvere i modelli che scriviamo: CPLEX, che risolve modelli di *Mathematical Programming*, e CP, che risolve modelli di *Constraint Programming*. CP ci permette risolvere modelli molto più complessi rispetto a CPLEX con lo svantaggio che ci mette molto più tempo solo per trovare una soluzione vicina a quella ottima, perché non fa ottimizzazioni come considerare la convessità dello spazio ammissibile, cutting plane o rilassamenti [3], e richiede tantisissimo tempo per stabilire se la f.o. trovata è quella ottima (per questo motivo conviene limitare il tempo di esecuzione).

Codice Per correttezza mostro brevemente il lavoro svolto. Sotto si possono osservare gli elementi che i due modelli hanno in comune: sono i dati e i vincoli.

Figura 3.3: Dati e vincoli dei due modelli

```
// Input dati del problema
int nUnita = ...;
int nPeriodi = ...;

range I = 1..nUnita;
range T = 1..nPeriodi;

int D[T] = ...;


tuple dati_unita
{
    float k1;
    float k2;
    float k3;

    int Pmin;
    int Pmax;

    int initial_status;

    int min_switch_up;
    int min_switch_down;

    int startup_cost;
}
dati_unita unita[I] = ...;

// Variabili
dvar boolean u[I][T];
dvar int p[I][T];
```

```
nUnita = 10;
nPeriodi = 24;

D = [800, 850, 950, 1050, 1100, 1200, 1250, 1300, 1400, 1500, 1550, 1600,
     1500, 1400, 1300, 1150, 1100, 1200, 1300, 1500, 1400, 1200, 1000, 900];

unita = #[
     1  : <1000, 16.19, 0.00048, 150, 455,  8, 8, 8,  9000>,
     2  : <970,  17.26, 0.00031, 150, 455,  8, 8, 8, 10000>,
     3  : <700,  16.60, 0.00200,  20, 130, -5, 5, 5,  1100>,
     4  : <680,  16.50, 0.00211,  20, 130, -5, 5, 5,  1120>,
     5  : <450,  19.70, 0.00398,  25, 162, -6, 6, 6,  1800>,
     6  : <370,  22.26, 0.00712,  20,  80, -3, 3, 3,   340>,
     7  : <480,  27.74, 0.00079,  25,  85, -3, 3, 3,   520>,
     8  : <660,  25.92, 0.00413,  55,  55, -1, 1, 1,    60>,
     9  : <665,  27.27, 0.00222,  55,  55, -1, 1, 1,    60>,
    10 :  <670,  27.79, 0.00173,  55,  55, -1, 1, 1,    60>
]#;
```

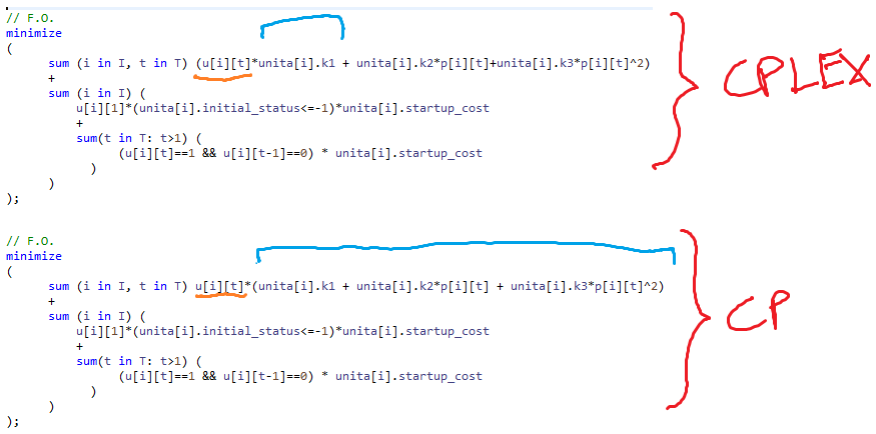
```
// VINCOLI
subject to
{
    // V.1
    forall (t in T)         vincolo_domanda_soddisfatta:            sum(i in I) p[i][t] == D[t];

    // V.5
    forall (i in I, t in T) vincolo_inf_potenza_prodotta_vincolata: (u[i][t]*unita[i].Pmin) <= p[i][t];
    // V.6
    forall (i in I, t in T) vincolo_sup_potenza_prodotta_vincolata: p[i][t] <= (u[i][t]*unita[i].Pmax);

    // V.9 (considerando V.11)
    vincolo_switch_up_minimo:
    forall (i in I, t in T)
        forall (r in t-unita[i].min_switch_up+1..t-1) {
                if (r<1)(u[i][t]>=
                    // (x <=|S|)*[(S>0)-(S<0)]+(S<0) with X=(-r+1) ... mi permette di ottenere la variabile decisionale u[i][r] con r non positivo
                    (((-r+1)<=abs(unita[i].initial_status))*((unita[i].initial_status>= 1)-(unita[i].initial_status<=-1))+(unita[i].initial_status<=-1))
                );
                if(r==1) (u[i][t] >= u[i][r]-(unita[i].initial_status>=1));
                if(r>1) (u[i][t] >= (u[i][r]-u[i][r-1]));
    }

    // V.10 (considerando V.11)
    vincolo_switch_down_minimo:
    forall (i in I, t in T)
            forall (r in t-unita[i].min_switch_down+1..t-1) {
                if(r<1) u[i][t] <= (1
                 -  (((-r+2)<=abs(unita[i].initial_status))*((unita[i].initial_status>= 1)-(unita[i].initial_status<=-1))+(unita[i].initial_status<=-1))
                 +  (((-r+1)<=abs(unita[i].initial_status))*((unita[i].initial_status>= 1)-(unita[i].initial_status<=-1))+(unita[i].initial_status<=-1))
                );
                if(r==1)u[i][t] <= (1 - (unita[i].initial_status>=1) + u[i][r]);
                if(r>1) u[i][t] <= (1 - u[i][r-1] + u[i][r]);
    }
}
```

La differenza fra i 2 modelli sta, oltre al fatto che nel modello CP bisogna specificare l'engine con la keyword *using CP;* e il tempo con *execute{cp.param.timeLimit=numOfSeconds;}*, nella f.o. La $c_i(u_{it}, p_{it}) = k_1(i)p_{it}^2 + k_2(i)p_{it} + k_3(i)$ ha valore non nullo solamente se $u[i][t] = 1$, di conseguenza feci la seguente formulazione: $c_i(u_{it}, p_{it}) = u[i][t] \times [k_1(i)p_{it}^2 + k_2(i)p_{it} + k_3(i)] = \boldsymbol{u[i][t]p_{it}p_{it}}k_1(i) + u[i][t]p_{it}k_2(i) + u[i][t]k_3(i)$. Ciò che non sapevo al tempo è che CPLEX supporta il prodotto di al massimo 2 variabili, mentre nell'espressione precedente abbiamo il prodotto di 3 variabili. I vincoli **V.5** e**V.6** ci dicono che se $u[i][t] = 0 \leftrightarrow p[i][t] = 0$, questo ci permette di riscrivere la formula precedente come $c_i(u_{it}, p_{it}) = p_{it}p_{it}k_1(i) + p_{it}k_2(i) + \boldsymbol{u[i][t]}k_3(i)$ che è finalmente una f.o. eseguibile su CPLEX.



Risultati La soluzione ottima di [1] è 610.646,5$, mentre i valori trovati dai miei modelli differiscono per un valore impercettibile ($\simeq 3$$).

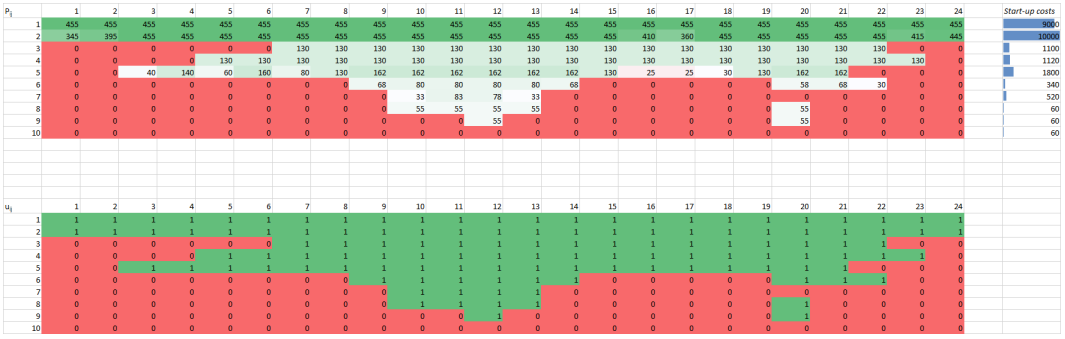
Modello CP		
f.o.	**Δ%**	**tempo**
611.292,454	0,105782	10s
610.973,460	0,053543	30s
610.711,743	0,010684	3m
610.711,743	0,010684	5m
610.711,743	0,010684	10m
610.649,461	0,000485	30m

Tabella 3.1: Risultati del modello CP: funzione obiettivo calcolata, differenza in percentuale col valore del paper e tempo impiegato

Modello CPLEX		
f.o.	**Δ%**	**tempo**
610.649,461	0,000485	0.22s

Tabella 3.2: Risultati del modello CPLEX: funzione obiettivo calcolata, differenza in percentuale col valore del paper e tempo impiegato

Come predetto nella precedente nota: le unità più grandi e con *SUc* elevati, che erano già attive all'inizio del problema, sono state tenute attive per tutto il tempo; le unità medie sono state avviate quando la domanda era un po' più alta, cioè all'inizio della mattina e sono state disattivate di notte; le unità piccole sono state avviate durante i picchi della domanda e tenute attive il meno possibile.



Bibliografia

[1] Alberto Borghetti et al. ≪Lagrangian relaxation and Tabu Search approaches for the unit commitment problem≫. In: vol. 3. Feb. 2001, 7 pp. vol.3. ISBN: 0-7803-7139-9. DOI: 10.1109/PTC.2001.964914.

[2] V. PETRIDIS, Spyros Kazarlis e Anastasios Bakirtzis. ≪Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problems≫. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 28 (nov. 1998), pp. 629–640. DOI: 10.1109/3477.718514.

[3] *Mathematical programming versus constraint programming — IBM® Decision Optimization CPLEX® Modeling for Python (DOcplex) V2.23 documentation*. https://ibmdecisionoptimization.github.io/docplex-doc/mp_vs_cp.html.