Algoritmi di ascesa duale per problemi di Facility Location

Alessio Di Luzio
Corso di Algoritmi e Modelli di Ottmizzazione Discreta
Università di "Tor Vergata"
Roma, Italia
alessio.diluzio94@gmail.com

I. INTRODUZIONE

I problemi di facility location affrontati sono:

A. *Uncapacitated Facility Location Problem*

Ogni cliente di un insieme V deve essere assegnato ad una ed una sola facility di un insieme U minimizzando i costi di connessione C e di apertura F delle facility.
Le variabili usate sono

$$y_{uv} = \begin{cases} 1 \; se \; il \; cliente \; v \; \text{è} \; assegnato \; alla \; facility \; u \\ 0 \; altrimenti \end{cases}$$

$$x_u = \begin{cases} 1 \; se \; la \; facility \; u \; \text{è} \; aperta \\ 0 \; altrimenti \end{cases}$$

I vincoli del problema impongono che ogni cliente v venga assegnato ad una ed una sola facility, che deve essere stata aperta.
Per questo problema sono stati confrontati i dati di risoluzione per la formulazione intera (PL 1)

$$\begin{array}{lll} \text{(PLI)} & \min & (\sum_{u\in U}\sum_{v\in V} c_{uv}y_{uv} + \sum_{u\in U} f_u x y_u) \\ & \text{s.t} & \sum_{u\in U} y x_{uv} = 1 \quad \forall\, v \in V \\ & & y_{uv} \le x_u \qquad \forall u \in U, v \in V \\ & & y_{uv} \in \{0,1\} \qquad \forall u \in U, v \in V \\ & & x_u \in \{0,1\} \qquad \forall u \in U \end{array}$$

PL 1. formulazione forte intera del UFL

e quelli ottenuti per il rilassamento lineare (PL 2)

$$\begin{array}{lll} \text{(PL)} & \min & (\sum_{u\in U}\sum_{v\in V} c_{uv}y_{uv} + \sum_{u\in U} f_u x_u) \\ & \text{s.t} & \sum_{u\in U} y_{uv} \ge 1 \qquad \forall\, v \in V \\ & & y_{uv} \le x_u \qquad \forall u \in U, v \in V \\ & & (x_u \le 1 \qquad \forall u \in U) \\ & & y_{uv} \ge 0, x_u \ge 0 \quad \forall u \in U, v \in V \end{array}$$

PL 2. formulazione forte rilassata del UFL

(entrambi calcolati sfruttando i software *AMPL* e *cplex*)
con quelli derivati dall'esecuzione di algoritmi euristici di ascesa duale; il *DUALOC* di Erlenkotter [1] e l'algoritmo *Primale-Duale* di [2] che sfruttano le proprietà del duale del rilassamento lineare (PL 3).
Il vincolo $x_u \le 1 \; \forall u \in U$ è inserito tra parentesi in quanto, per la soluzione ottima, è implicato dalla natura di minimizzazione della funzione obiettivo (come del resto il vincolo $y_{uv} \le 1$ che invece è stato omesso), tuttavia la relativa variabile duale è usata nella formulazione del problema PL 3.

$$\begin{array}{lll} \text{(D)} & \max & (\sum_{v\in V} z_v - \sum_{u\in U} t_u) \\ & \text{s.t.} & z_v - w_{uv} \le c_{uv} \qquad \forall u \in U, v \in V \\ & & \sum_{v\in V} w_{uv} - t_u \le f_u \quad \forall u \in U \\ & & z_v \ge 0, t_u \ge 0, w_{uv} \ge 0 \; \forall u \in U, v \in V \end{array}$$

PL 3. formulazione duale di PL 2

La variabile duale t_u è relativa al vincolo primale $x_u \le 1 \; \forall u \in U$, come si può notare la soluzione ottima del duale *W(D)* avrà il vettore t a componenti nulle rendendo quindi 'trascurabile' il vincolo primale, come affermato.

Entrambi gli algoritmi duali in ogni iterazione calcolano un incremento Δ per la soluzione corrente, il *DUALOC* incrementa le singole variabili z_v prese una alla volta finché non è più possibile calcolare un Δ strettamente positivo. L'*Algoritmo Primale – Duale* invece incrementa, in una singola iterazione, tutte le variabili z_v finché, anche in questo caso, non è possibile calcolare un Δ strettamente positivo.
A differenza del *DUALOC* nell'algoritmo *Primale – Duale* ad ogni iterazione si considerano anche le 'conseguenze' sul problema primale; una variabile z_v smette di essere incrementata quando il cliente v è assegnato a una facility aperta; una facility u è dichiarata temporaneamente aperta quando il vincolo $\sum_{v\in V} w_{uv} - t_u \le f_u$ è soddisfatto all'uguaglianza, l'algoritmo si arresta quando tutti i clienti sono stati assegnati; sono assegnati alla facility u tutti i clienti v ,non ancora collocati, per cui il vincolo $z_v - w_{uv} \le c_{uv}$ è soddisfatto all'uguaglianza.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

B. Single Source Capacitated Facility Location Problem

Una variante del problema UFL in cui ogni cliente v ha una domanda d_v ed ogni facility u una capacità K_u.

Come per il Simple Facility Location sono stati confrontati i risultati ottenuti con il solver software *AMPL* per il problema primale intero (PL 4) e rilassato (PL 5) con quelli derivati dall'esecuzione di un algoritmo di ascesa duale.

$$
\begin{array}{lll}
\text{(PLI)} & \min & (\sum_{u\in U}\sum_{v\in V} c_{uv}y_{uv} + \sum_{u\in U} f_u x_u) \\
& \text{s.t} & \sum_{u\in U} y_{uv} = 1 \qquad \forall\, v \in V \\
& & \sum_{v\in V} d_v\, y_{uv} \le K_u x_u \qquad \forall u \in U, v \in V \\
& & y_{uv} \in \{0,1\} \qquad \forall u \in U, v \in V \\
& & x_u \in \{0,1\} \qquad \forall u \in U
\end{array}
$$

PL 4. formulazione intera del SSCFL

$$
\begin{array}{lll}
\text{(PL)} & \min & (\sum_{u\in U}\sum_{v\in V} c_{uv}y_{uv} + \sum_{u\in U} f_u x_u) \\
& \text{s.t} & \sum_{u\in U} y_{uv} \ge 1 \qquad \forall\, v \in V \\
& & \sum_{v\in V} d_v\, y_{uv} \le K_u x_u \qquad \forall u \in U, v \in V \\
& & x_u \le 1 \qquad \forall u \in U \\
& & x_u \ge 0, y_{uv} \ge 0 \qquad \forall u \in U, v \in V
\end{array}
$$

PL 5. formulazione rilassata del SSCFL

In questo caso il vincolo $x_u \le 1$ non è 'trascurabile', potrebbe infatti essere vantaggioso ingrandire una facility (portandone ovvero la relativa x_u oltre 1) per aumentarne la capacità e soddisfare una maggiore domanda.
L'algoritmo *DUALOC* è stato modificato per adattarlo al problema di SSCFL seguendo l'analogo ragionamento della trattazione di [1] pagine 288-304.

$$
\begin{array}{lll}
\text{(D)} & \max & (\sum_{v\in V} z_v - \sum_{u\in U} t_u) \\
& \text{s.t.} & z_v - w_u d_v \le c_{uv} \qquad \forall u \in U, v \in V \\
& & q_u w_u - t_u \le f_u \qquad \forall u \in U \\
& & z_v \ge 0, t_u \ge 0, w_u \ge 0 \; \forall u \in U, v \in V
\end{array}
$$

PL 6. formulazione duale di PL 5

II. Realizzazione

Tutti i risultati sono stati ottenuti sviluppando un software in Python.
Nel programma è compresa l'implementazione degli algoritmi approssimati di ascesa duale (*DUALOC,Primale-Duale,DUALOC modificato per SSCFL*) e l'integrazione con le API AMPL per la risoluzione dei problemi usando il solver *AMPL*.

A. DUALOC

Il *DUALOC* implementato è quello presente in forma di pseudocodice alle pagine 303-304 di [1].

1 *Inizializzazione:* $\overline{z_v} = min_{u\in U}\{c_{uv}\}\; \forall\, v \in V\,, i := 1;$
2 *Passo (i)*
3 $\bar{V} = V;$
4 ***Ripeti***
5 $\quad \bar{v} = argmin_{v\in \bar{V}}\{h(v)\};$
6 $\quad \bar{V} := \bar{V} - \{\bar{v}\}\,;$
7 $\quad \tau_{\bar{v}u} = f_u - \sum_{v\in V-\{\bar{v}\}} max\{0, \overline{z_v} - c_{uv}\};$
8 $\quad b_{\bar{v}} = min_{u\in U}\{c_{u\bar{v}} + \tau_{\bar{v}u} - \overline{z_{\bar{v}}}\};$
9 $\quad b'_{\bar{v}} = min\left\{b_{\bar{v}}, min_{\substack{u\in U \\ c_{u\bar{v}} - \overline{z_{\bar{v}}} > 0}}\{c_{u\bar{v}} - \overline{z_{\bar{v}}}\}\right\};$
10 **fino a che** $b'_{\bar{v}} > 0$ **oppure** $\bar{V} = \emptyset$
11 **Se** $\boldsymbol{b'_{\bar{v}} = 0} \Rightarrow STOP.$
12 **Altrimenti**
13 $\quad \overline{z_{\bar{v}}} := \overline{z_{\bar{v}}} + b'_{\bar{v}}\,;$
14 $\quad i := i + 1;$
15 **Vai** al Passo *(* i);

Algoritmo 1. DUALOC di Erlenkotter per UFL

B. Algoritmo Primale-Duale

L'implementazione segue pedissequamente quanto esposto in [2] alla pagina 7.

C. DUALOC adattato a SSCFL

Con un ragionamento analogo a quello che,nella trattazione di [1] alle pagine 288-304, ha portato all'algoritmo DUALOC, si possono apportare delle modifiche per adattarlo al problema capacitato.
L'obiettivodi ogni iterazione, anche in questo caso, è incrementare una delle variabili $\overline{z_{\bar{v}}}$ aumentando complessivamente il valore della funzione obiettivo e contemporaneamente mantenere il vettore t a componenti nulle.
Il vettore t per ogni facility u deve rispettare il vincolo $t_u \ge q_u w_u - f_u$, quindi per minimizzare t_u occorre minimizzare w_u che a sua volta deve obbedire al vincolo $w_u \ge (z_v - c_{uv}) * \frac{1}{d_v}$.
Scegliamo quindi

$$\overline{w_u} = argmax_{v\in V}\left\{max\left\{0, (z_v - c_{uv}) * \frac{1}{d_v}\right\}\right\}$$

(con l'operazione di massimo come al solito utilizzata per garantire la non negatività delle variabili w_u).

In modo analogo $t_u = max\{0, q_u\overline{w_u} - f_u\}$ cioè

$$t_u = max\left\{0, q_u * argmax_{v\in V}\left\{max\left\{0, (z_v - c_{uv}) * \frac{1}{d_v}\right\}\right\} - f_u\right\}$$

Da ciò è evidente che inizializzando $\bar{z_v} = min_{u\in U}\{c_{uv}\}$ si ottiene un vettore t iniziale a componenti nulle.

Preso quindi un cliente $\bar{v} \in \bar{V}$ dobbiamo calcolare il Massimo incremento $\Delta_{\bar{v}}$ per la relativa variabile $\bar{z_{\bar{v}}}$ che mantenga il vettore t a componenti nulle.
Ovvero occorre garantire che per ogni facility u

$$q_u * argmax_{v\in V}\left\{max\left\{0, (z_v - c_{uv}) * \frac{1}{d_v}\right\}\right\} - f_u \leq 0$$

$$q_u * (\bar{z_{\bar{v}}} + \Delta_{\bar{v}} - c_{uv}) * \frac{1}{d_v} - f_u \leq 0$$

$$\Delta_{\bar{v}} \leq \frac{q_u}{d_v} * f_u - z_v + c_{uv}$$

Si passa dalla prima alla seconda delle precedenti equazioni considerando che il $\Delta_{\bar{v}}$ massimo renderebbe il cliente $\bar{v}$ 'vincitore' dell'operazione di argmax presente nella prima equazione.
Si ottiene $\Delta_{\bar{v}}$ garantendo che ogni componente t_u di t sia nulla ovvero $\Delta_{\bar{v}} = min_{u\in U}\left\{\frac{q_u}{d_v} * f_u - z_v + c_{uv}\right\}$.

L'algoritmo può essere così modificato

1 *Inizializzazione:* $\bar{z_v} = min_{u\in U}\{c_{uv}\}\ \forall\ v \in V$
2 $\bar{V} = V$;
3 ***Ripeti***
4 $\bar{v} = argmin_{v\in\bar{V}}\{h(v)\}$;
5 $\bar{V} := \bar{V} - \{\bar{v}\}$;
6 $b_{\bar{v}} = min_{u\in U}\left\{\frac{q_u}{d_v} * f_u - z_v + c_{uv}\right\}$;
7 $\bar{z_{\bar{v}}} := \bar{z_{\bar{v}}} + b_{\bar{v}}$;
8 **fino a che** $\bar{V} = \emptyset$

Algoritmo 2. DUALOC modificato per SSCFL

Nell'Algoritmo 2 è stata omessa la riga 8 del DUALOC originale ($b'_{\bar{v}} = min\left\{b_{\bar{v}}, min_{\substack{u\in U \\ c_{u\bar{v}} - \bar{z_{\bar{v}}} > 0}}\{c_{u\bar{v}} - \bar{z_{\bar{v}}}\}\right\}$) questo perchè eseguendolo sulle istanze di test si è notato come non ci fossero sensibili miglioramenti nel valore calcolato per la funzione obiettivo a fronte però di un pesante rallentamento nell'esecuzione.

D. ASCESA DUALE SEMPLICE

L'algoritmo di ascesa duale 'semplice' è quello riportato a pagina 297 di [1].

1 *Inizializzazione:* $\bar{z_v} = min_{u\in U}\{c_{uv}\}\ \forall\ v \in V$
2 ***Ripeti*** $\forall\ \bar{v} \in \boldsymbol{V}$
3 $\tau_{\bar{v}u} = f_u - \sum_{v\in V-\{\bar{v}\}} \max\{0, \bar{z_v} - c_{uv}\}$;
4 $b_{\bar{v}} = min_{u\in U}\{c_{u\bar{v}} + \tau_{\bar{v}u} - \bar{z_{\bar{v}}}\}$;
5 $\bar{z_{\bar{v}}} := \bar{z_{\bar{v}}} + b_{\bar{v}}$;
6 **end;**

Questo algoritmo è stato implementato nel software Python per comparare la qualità della soluzione trovata con quella del *DUALOC* citato in precedenza.

III. RISULTATI DI TESTING

Sono state considerate 30 istanze di problemi di SSCFL (per analizzare l'UFL sono stati ovviamente ignorati i valori di domanda e capacità).
L'algoritmo *Primale-Duale* lavora 'bene' su istanze che rispettino l'ipotesi metrica e per studiarne il comportamento sono state modificate 5 istanze ,tra le 30 analizzate, per renderle 'metriche'.
Tutti i grafici presenti nei paragrafi che seguono sono stati ottenuti con l'utilizzo del software MATLAB.

In tutti i grafici che seguono nell'asse delle ascisse è misurato un valore indicativo della 'dimensione' dell'istanza , in questo è stato considerato il prodotto $|V| * |U|$ del numero di clienti e di facility.

A. UFL

I risultati del testing mostrati nelle tabelle allegate confermano quanto affermato alla pagina 304 di [1].
L'algoritmo *DUALOC* ha prodotto dei Bound di ottima qualità (in molti casi ottimi) in modo efficiente, risolvendo il problema in un tempo tuttavia alle volte sensibilmente superiore a quello impiegato dal solver *AMPL*.
I bound forniti dall'algoritmo *Primale-Duale* sono sempre non migliori di quelli calcolati dal *DUALOC* e sempre a fronte di un netto degrado delle prestazioni.

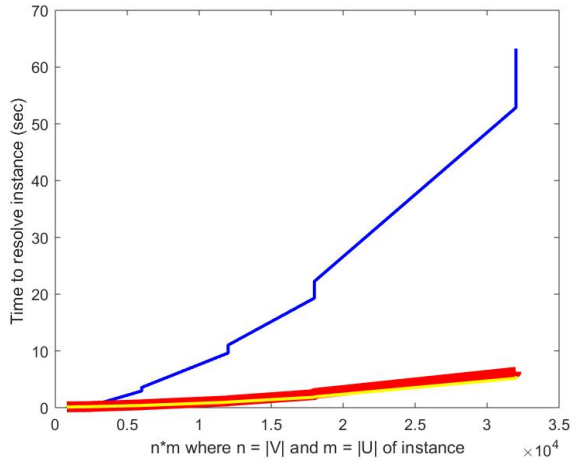

Figura 1. Confronto dei tempi di esecuzione del solver AMPL sul problema in forma intera (linea rossa), in forma rilassata (linea gialla) e dell'algoritmo *DUALOC* (linea blu).

Come evidenziato dalla Figura 1, sulle 30 istanze considerate, il tempo di esecuzione dell'algoritmo *DUALOC* è sensibilmente peggiore di quello del solver AMPL sul problema intero e su quello rilassato.
Tuttavia, in Figura 2, si può notare come l'algoritmo approssimato abbia fornito una soluzione di ottima qualità.

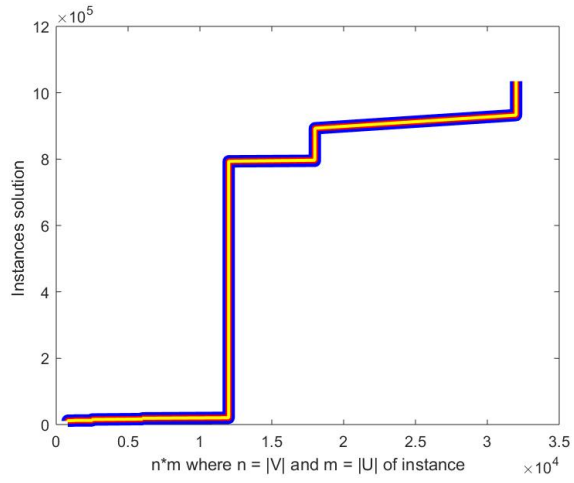

Figura 2. Confronto del valore della soluzione trovata dal solver AMPL sul problema in forma intera (linea rossa), in forma rilassata (linea gialla) e dall'algoritmo *DUALOC* (linea blu).

La buona qualità della soluzione calcolata con il *DUALOC* permette all'euristica descritta alle pagine 304-306 di [1] di fornire anche una buona soluzione per il problema primale, come mostrato in Figura 3.

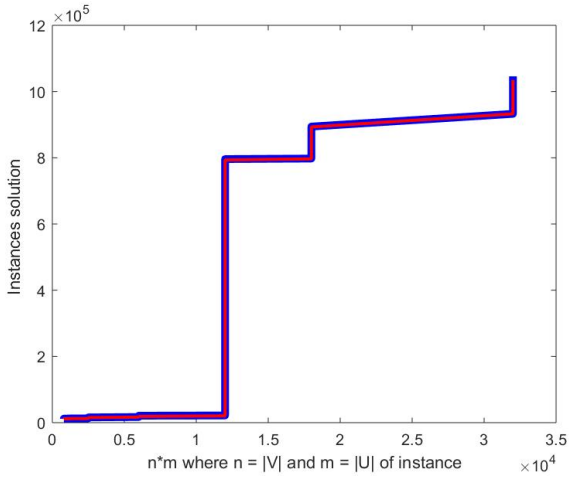

Figura 3. Confronto del valore della soluzione trovata dal solver AMPL sul problema in forma intera (linea rossa) con la soluzione primale individuate dall'algoritmo *DUALOC* (linea blu) con l'idea euristica di pagine 304-306 in [1]

Nell'algoritmo *Primale–Duale*, per costruzione, ad ogni iterazione il valore di incremento Δ viene calcolato, almeno inizialmente, tenendo conto di tutti i vincoli del problema originale con la conseguenza di una covergenza molto rallentata rispetto al *DUALOC* (Figura 4).

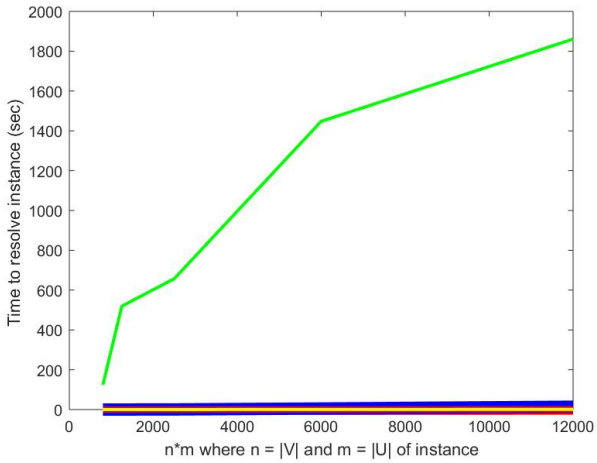

Figura 4. Confronto dei tempi di esecuzione del solver AMPL sul problema in forma intera (linea rossa), in forma rilassata (linea gialla),dell'algoritmo *DUALOC* (linea blu) e dell'algoritmo *Primale–Duale* (linea verde) su istanze di tipo metrico.

Sebbene le prestazioni dell'algoritmo *Primale-Duale* siano peggiori di quelle del *DUALOC*, il valore della soluzione è,anche in questo caso di ottima qualità (Figura 5).

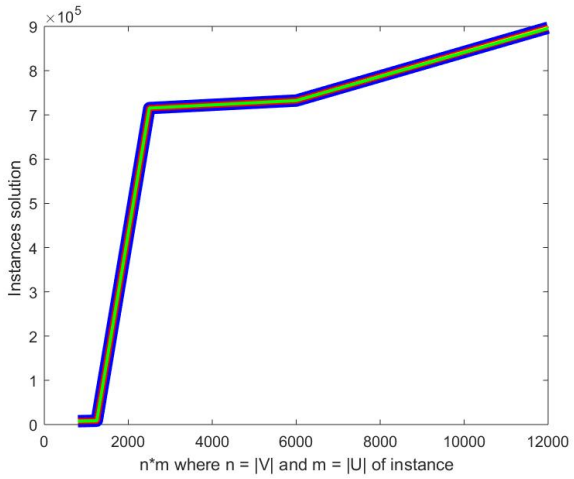

Figura 5. Confronto del valore della soluzione trovata dal solver AMPL sul problema in forma intera (linea rossa), dall'algoritmo *DUALOC* (linea blu) e dall'algoritmo *Primale–Duale* (linea verde).

Infine,le Figure 6 e 7 mostrano come l'algoritmo *DUALOC* calcoli un Lower Bound più preciso rispetto all'algoritmo di Ascesa Duale semplice a discapito di un degrado delle prestazioni.

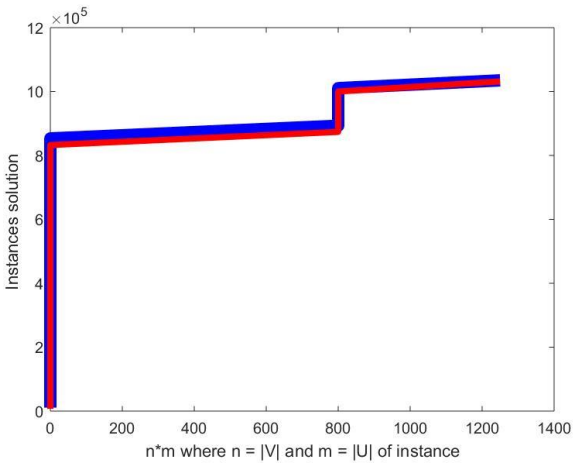

Figura 6. Confronto del valore della soluzione travata dall'algoritmo *DUALOC* (linea blu) e dall'algoritmo di Ascesa Duale semplice (linea rossa).

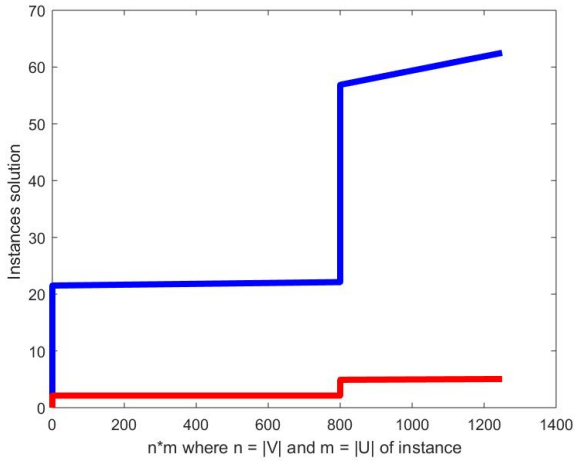

Figura 7. Confronto dei tempi di esecuzione dell'algoritmo *DUALOC* (linea blu) e dell'algoritmo di Ascesa Duale semplice (linea rossa).

B. SSCFL

Anche in questo caso la soluzione del *DUALOC* è di buona qualità (cioè molto vicina a quella della formulazione rilassata).
Il tempo impiegato dall'algoritmo approssimato è notevolmente inferiore a quello servito al solver *AMPL* per risolvere il problema intero, ma maggiore di quello impiegato nella soluzione del problema rilassato.

In questo caso non sono state calcolate ipotetiche soluzioni primali, in quanto anche ammettendo di riuscire a individuare un insieme di facility candidate ad essere aperte,la scelta di come connettervi i client è non banale; si potrebbe risolvere un problema di knapsack binario per connettere i primi k clienti alla prima facility e poi continuare con i restanti $n - k$ clienti e le restanti facility fino ad assegnare tutti i clienti.

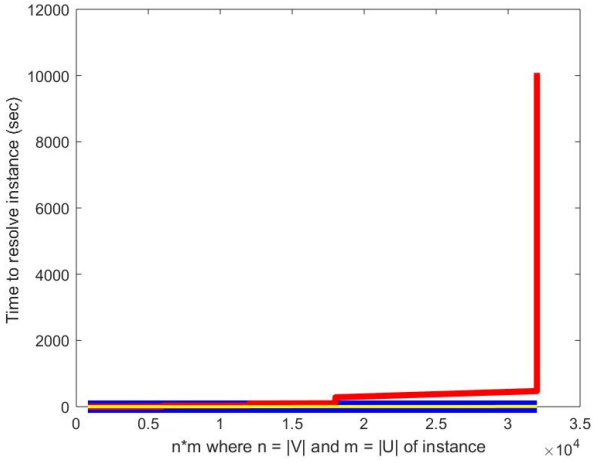

Figura 8. Confronto dei tempi di esecuzione del solver AMPL sul problema in forma intera (linea rossa), in forma rilassata (linea gialla) e dell'algoritmo *DUALOC* adattato (linea blu).

La Figura 8 mostra come, per le istanze esaminate, l'algoritmo *DUALOC* adattato al problema SSCFL abbia delle ottime prestazioni, paragonabili a quelle del solver AMPL per il problema rilassato, e nettamente migliori di quelle del solver sul problema intero.

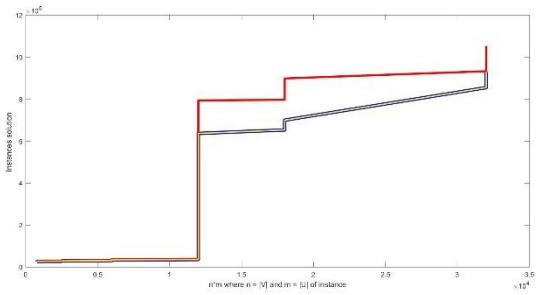

Figura 9. Confronto del valore della soluzione trovata dal solver AMPL sul problema in forma intera (linea rossa), in forma rilassata (linea gialla) e dall'algoritmo *DUALOC* adattato (linea blu).

Infine in Figura 9. vediamo come la soluzione fornita dal *DUALOC* adattato, sebbene si discosti sensibilmente da quella del problema intero, è aderente al Lower Bound calcolato risolvendo il rilassamento lineare del SSCFL.

IV. MANUALE

Il software può essere eseguito su un qualunque SO,ma è necessario che vi siano installati una realease di Python 3 e che vi sia presente il software AMPL.

E' inoltre necessario installare i package Python amply (API AMPL per Python) e prettytable (per il rendering in forma tabulare dei risultati di test)

Per installare questi package è sufficiente eseguire i comandi

```
python -m pip install pip --upgrade
python -m pip install amplpy
python -m pip install prettytable
```

Per avviare il software occorre aprire un terminale nella cartella del progetto e eseguire il commando

```
python main.py
```

Un menù interattivo guiderà nella scelta dell'operazione da eseguire, in particolare verrà chiesto

- Se calcolare la soluzione di problemi UFL o SSCFL
- Se calcolare la soluzione di una istanza o più istanze, nel secondo caso è necessario fornire il path a una cartella che deve contenere solamente i file di istanza.
- Se avere in ouput un file in formato .csv e uno in .txt con i risultati dell'esecuzione.
- Se mostrare i singoli step di esecuzione degli algoritmi.
- Se calcolare anche la soluzione primale relativa alle esecuzioni di *DUALOC* e algoritmo *Primale-Duale* per l'UFL.

Sarà richiesto il path alla directory del software AMPL se questa non fosse già presente nella variabile PATH di Sistema.

RIFERIMENTI

[1] A. Sassano, Modelli e algoritmi della ricerca operativa, FrancoAngeli, 1999, Milano, pp. 267 – 308.

[2] A. Gupta,V. Gupta , "Lecture 5: Primal-Dual Algorithms and Facility Location", 2008, pp. 1 - 7.