



# **Analisi prestazionale: problemi di facility location**

Implementazione dell'algoritmo di ascesa duale e confronto tra rilassamento lagrangiano e rilassamento lineare.

Luca Fiscariello  
0318266

---

# Obiettivi

- Implementazione dell'algoritmo di ascesa duale per un problema di facility location nella versione capacitata e non capacitata.
- Valutazione delle prestazioni degli algoritmi implementati.
- Individuazione del miglior lower bound calcolato tramite rilassamento lineare e rilassamento lagrangiano.

---

# Metodologia

- Scrittura del modello “manuale”.
- Trascrizione del modello su un solver commerciale. In particolare è stato utilizzata l'API python offerta da **Gurobi**.
- Generazione dei casi di test sfruttando un approccio statistico.
- Generazione di grafici per interpretare i risultati.

# Scrittura del modello

I due modelli su cui si basa il progetto sono: il problema di facility location capacitato (CFL) e il problema di facility location non capacitato(UFL). Per ciascuno di questi due modelli sono state implementate diverse varianti del problema:

CFL	UFL
Modello base	Modello base
Modello rilassato linearmente	Modello rilassato linearmente
Modello rilassato lagrangeanamente	Modello rilassato lagrangeanamente
Algoritmo di ascesa duale	Algoritmo di ascesa duale

# Scrittura del modello : UFL

$$\min \sum_{u \in S} X_u F_u + \sum_{u \in S} \sum_{v \in D} Y_{uv} C_{uv}$$

```
# Modello
m = gp.Model("facility")
m.ModelSense = GRB.MINIMIZE
m.Params.Method = 2
```

$$\sum_{u \in S} Y_{uv} = 1 \quad \forall v \in D$$

```
# Vincoli sul soddisfacimento della domanda dei clienti
# Sommatoria su u (yuv) == 1
m.addConstrs((y.sum('*', v) == 1 for v in clients), "Demand")
```

$$X_u - Y_{uv} \geq 0 \quad \forall uv \in S \times D$$

```
# Vincoli che legano le variabili di xu e yuv
# yuv<=xu
m.addConstrs((x[u]-y[u, v] >= 0 for u in facility for v in clients), "Continuity")
```

$$X_u, Y_{uv} \in \{0,1\}$$

```
# Variabile decisionale: x[u] == 1 se centro u è attivo.
x = m.addVars(facility, vtype=GRB.BINARY, obj=setup_costs, name="X")
```

```
# Variabile decisionale : y[u][v] == 1 se centro u serve cliente v
y = m.addVars(facility, clients, vtype=GRB.BINARY, obj=allocation_costs, name="Y")
```

# Scrittura del modello : UFL lagrange

Vincoli rilassati :  $X_u - Y_{uv} \geq 0 \quad \forall uv \in S \times D$

$$\min \sum_{u \in S} X_u F_u + \sum_{u \in S} \sum_{v \in D} Y_{uv} C_{uv} - \sum_{u \in S} \sum_{v \in D} \lambda_{uv} (X_u - Y_{uv})$$

$$\min \sum_{u \in S} X_u \left( F_u - \sum_{v \in D} \lambda_{uv} \right) + \sum_{u \in S} \sum_{v \in D} Y_{uv} \left( C_{uv} + \lambda_{uv} \right)$$

```
x_moltiplicator = [self.setup_costs[u] - sum(lambda_vector) for u in facility]
```

```
y_moltiplicator = [self.allocation_costs[(self.client_number+1)*v + u] + lambda_vector[(self.client_number+1)*v + u] for v in clients for u in facility]
```

Per ogni rilassamento sono state provate **20 combinazioni diverse** di moltiplicatori. Successivamente è stato preso in analisi solo il rilassamento di valore massimo.

# Scrittura del modello : UFL ascesa duale (1/2)

Primale	Duale	Generalizzazione
$X \equiv \{X_a, X_b, X_c\}$		$Z_1 \leq C_{a1} + w_{a1}$
$Y \equiv \{Y_{a1}, Y_{a2}, Y_{b1}, Y_{b2}, Y_{c1}, Y_{c2}\}$	$\max Z_1 + Z_2$	$Z_1 \leq C_{b1} + w_{b1}$
$Y_{a1} + Y_{b1} + Y_{c1} = 1 \rightarrow Z_1$	$w_{a1} + w_{a2} \leq F_a$	$Z_1 \leq C_{c1} + w_{c1}$
$Y_{a2} + Y_{b2} + Y_{c2} = 1 \rightarrow Z_2$	$w_{b1} + w_{b2} \leq F_b$	$Z_1 = \operatorname{argmin}(C_{a1} + w_{a1}, C_{b1} + w_{b1}, C_{c1} + w_{c1})$
$X_a - Y_{a1} \geq 0 \rightarrow w_{a1}$	$w_{c1} + w_{c2} \leq F_c$	$w_{a1} = \max(0, F_a - w_{a2})$
$X_a - Y_{a2} \geq 0 \rightarrow w_{a2}$	$Z_1 \leq C_{a1} + w_{a1}$	$w_{b1} = \max(0, F_b - w_{b2})$
$X_b - Y_{b1} \geq 0 \rightarrow w_{b1}$	$Z_2 \leq C_{a2} + w_{a2}$	$w_{c1} = \max(0, F_c - w_{c2})$
$X_b - Y_{b2} \geq 0 \rightarrow w_{b2}$	$Z_1 \leq C_{b1} + w_{b1}$	$Z_v = \operatorname{argmin}_{u \in S}(C_{uv} + w_{uv})$
$X_c - Y_{c1} \geq 0 \rightarrow w_{c1}$	$Z_2 \leq C_{b2} + w_{b2}$	$w_{uv} = \max(0, F_u - \sum_{k \in D} w_{uk} + w_{uv})$
$X_c - Y_{c2} \geq 0 \rightarrow w_{c2}$	$Z_1 \leq C_{c1} + w_{c1}$	
	$Z_2 \leq C_{c2} + w_{c2}$	

# Scrittura del modello : UFL ascesa duale (2/2)

Per implementare l'algoritmo è necessario inizializzare alcuni valori:

Inizializzo  $Z_v : \min_{u \in S} (C_{uv})$

Inizializzo  $w_{uv} : \max(0, F_u - C_{uv})$

La rispettiva implementazione:

```
#Inizializzo zv al minimo di cuv scorrendo tutte le u
for v in range(len(c[0])):
    temp = []
    for u in range(len(c)):
        temp.append(c[u][v])
    z.append(min(temp))

#calcolo wuv=max(0,zv - cuv)
for u in range(len(c)):
    temp = []
    for v in range(len(c[0])):
        temp.append(max(0, z[v]-c[u][v]))
    w.append(temp)
```

Inizializzati i valori si procede con l'esecuzione:

$$Z_v = \operatorname{argmin}_{u \in S} (C_{uv} + w_{uv})$$
$$w_{uv} = \max(0, F_u - \sum_{k \in D} w_{uk} + w_{uv})$$

La rispettiva implementazione:

```
#calcolo deltauv
for u in range(len(c)):
    temp = []
    for v in range(len(c[0])):
        temp.append(f[v] - (sum_w - w[u][v]))
    delta.append(temp)

#trovo valori finali di zv come il minimo z[v] + delta[u][v] scorrendo le u
for v in range(len(c[0])):
    temp = []
    for u in range(len(c)):
        temp.append(z[v] + delta[u][v])
    z_final.append(min(temp))
```

# Scrittura del modello : CFL

$$\min \sum_{u \in S} X_u F_u + \sum_{u \in S} \sum_{v \in D} Y_{uv} C_{uv}$$



```
# Modello
m = gp.Model("facility")
m.ModelSense = GRB.MINIMIZE
m.Params.Method = 2
```

$$\sum_{u \in S} Y_{uv} = 1 \quad \forall v \in D$$



```
# Vincoli sulla capacità
# Sommatoria su v (yuv)*dv <= ku*xu
m.addConstrs((sum(y[u, v]*demands[v] for v in clients) <= capacity[u] * x[u] for u in facility), "Capacity")
```

$$\sum_{v \in D} Y_{uv} d_v \leq X_u k_u \quad \forall u \in S$$



```
# Vincoli sul soddisfacimento della domanda dei clienti
# Sommatoria su u (yuv) == 1
m.addConstrs((y.sum('*', v) == 1 for v in clients), "Demand")
```

$$X_u, Y_{uv} \in \{0,1\}$$



```
# Variabile decisionale: x[u] == 1 se centro u è attivo.
x = m.addVars(facility, vtype=GRB.BINARY, obj=setup_costs, name="X")

# Variabile decisionale : y[u][v] == 1 se centro u serve cliente v
y = m.addVars(facility, clients, vtype=GRB.BINARY, obj=allocation_costs, name="Y")
```

# Scrittura del modello : UFL lagrange

Vincoli rilassati:  $\sum_{v \in D} Y_{uv} d_v \leq X_u k_u \quad \forall u \in S$

$$\min \quad \sum_{u \in S} X_u F_u + \sum_{u \in S} \sum_{v \in D} Y_{uv} C_{uv} - \sum_{u \in S} \lambda_u (X_u k_u - \sum_{v \in D} Y_{uv} d_v)$$

Per ogni rilassamento sono state provate **20 combinazioni diverse** di moltiplicatori. Successivamente è stato preso in analisi solo il rilassamento di valore massimo.

$$\min \quad \sum_{u \in S} X_u (F_u - \lambda_u k_u) + \sum_{u \in S} \sum_{v \in D} Y_{uv} (C_{uv} + d_v \lambda_u)$$



```
x_moltiplicator = [self.setup_costs[u] - self.capacity[u]*lambda_vector[u] for u in facility]
```

```
y_moltiplicator = [self.allocation_costs[(self.client_number+1)*v + u] + lambda_vector[u]+self.demands[v] for v in clients for u in facility]
```

# Scrittura del modello : CFL ascesa duale (1/2)

Primale	Duale	Generalizzazione
$X \equiv \{X_a, X_b, X_c\}$	$\max \lambda_1 + \lambda_2$	$\lambda_1 \leq C_{a1} + \gamma_a d_1$
$Y \equiv \{Y_{a1}, Y_{a2}, Y_{b1}, Y_{b2}, Y_{c1}, Y_{c2}\}$	$\gamma_a k_a \leq F_a$	$\lambda_1 \leq C_{b1} + \gamma_b d_1$
$Y_{a1} + Y_{b1} + Y_{c1} = 1 \rightarrow \lambda_1$	$\gamma_b k_b \leq F_b$	$\lambda_1 \leq C_{c1} + \gamma_c d_1$
$Y_{a2} + Y_{b2} + Y_{c2} = 1 \rightarrow \lambda_2$	$\gamma_c k_c \leq F_c$	$\lambda_1 = \operatorname{argmin} (C_{a1} + \gamma_a d_1, C_{b1} + \gamma_b d_1, C_{c1} + \gamma_c d_1)$
$X_a k_a - (Y_{a1}d_1 + Y_{a2}d_2) \geq 0 \rightarrow \gamma_a$	$\lambda_1 - \gamma_a d_1 \leq C_{a1}$	$\gamma_a \leq \frac{F_a}{k_a}, \gamma_b \leq \frac{F_b}{k_b}, \gamma_c \leq \frac{F_c}{k_c}$
$X_b k_b - (Y_{b1}d_1 + Y_{b2}d_2) \geq 0 \rightarrow \gamma_b$	$\lambda_2 - \gamma_a d_2 \leq C_{a2}$	$\lambda_v = \operatorname{argmin}_{u \in S} (C_{uv} + \gamma_u d_v)$
$X_c k_c - (Y_{c1}d_1 + Y_{c2}d_2) \geq 0 \rightarrow \gamma_c$	$\lambda_1 - \gamma_b d_1 \leq C_{b1}$	$\gamma_u \leq \frac{F_u}{k_u}$
	$\lambda_2 - \gamma_b d_2 \leq C_{b2}$	
	$\lambda_1 - \gamma_c d_1 \leq C_{c1}$	
	$\lambda_2 - \gamma_c d_2 \leq C_{c2}$	

# Scrittura del modello : CFL ascesa duale (2/2)

Due osservazioni:

- La domanda di un singolo cliente dovrebbe sempre essere più piccola della capacità di un centro.
- La soluzione del duale deve essere intera.

Combinando le osservazioni si ottiene:

$$\lambda_v = \operatorname{argmin}_{u \in S} \left( C_{uv} + \frac{F_u}{k_u} d_v \right) \longrightarrow \lambda_v = \operatorname{argmin}_{u \in S} \left( C_{uv} + \left\lceil \frac{F_u}{k_u} d_v \right\rceil \right)$$

La corrispondente implementazione :

```
#trovo valori finali di lv come c[u][v] + f[u] in corrispondenza del minimo c[u][v] + delta[u][v] scorrendo le u
for v in range(len(c[0])):
    temp = []
    temp_1 = []
    for u in range(len(c)):
        temp.append(c[u][v] + delta[u][v])
        temp_1.append(c[u][v] + f[u])

    min_value = min(temp)
    min_pos = temp.index(min_value)
    l_final.append(temp_1[min_pos])
```

---

## Generazione dei casi di test

I test sono stati generati seguendo due metodologie:

1. Fissata la dimensione del problema (numero di variabili decisionali) si fanno variare i parametri come i costi fissi associati a un centro o i costi di allocazione.
2. Si incrementa progressivamente la dimensione del problema e si analizzano le prestazioni.

L'analisi delle prestazioni riguarda la vicinanza alla soluzione ottima ma anche il tempo di esecuzione.



## Generazione dei casi di test

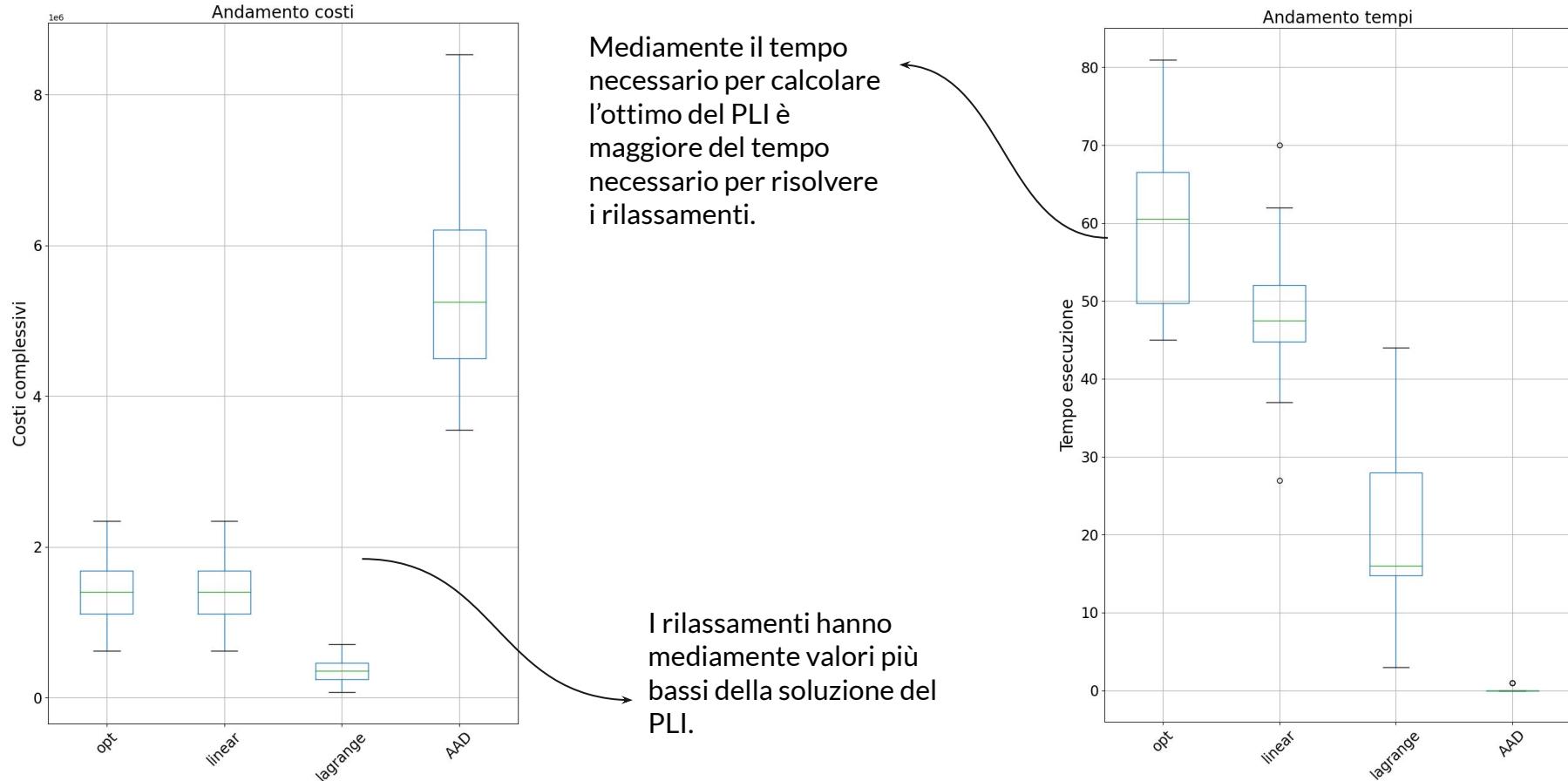
Come specificato in precedenza l'approccio applicato è statistico. I valori che verranno mostrati nei grafici non sono ottenuti da un'unica esecuzione, ma sono il risultato di più prove con parametri differenti. I parametri utilizzati nei test sono ottenuti da un **generatore pseudo-random multistream**.

Per ogni test, di entrambi gli approcci, e per entrambe le versioni del problema è stato calcolato:

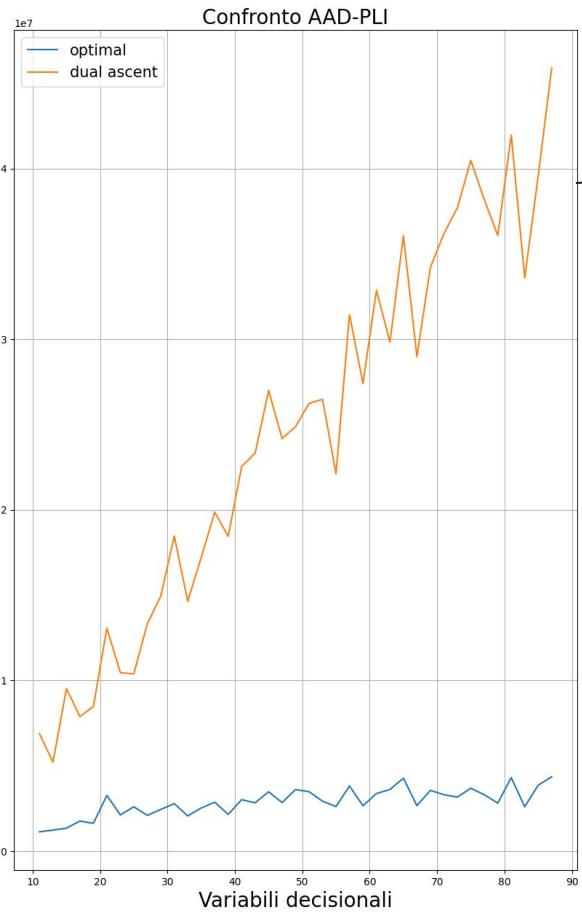
- soluzione ottima
- rilassamento lineare
- rilassamento lagrangeano
- soluzione AAD

I dati sono stati raccolti in opportuni CSV.

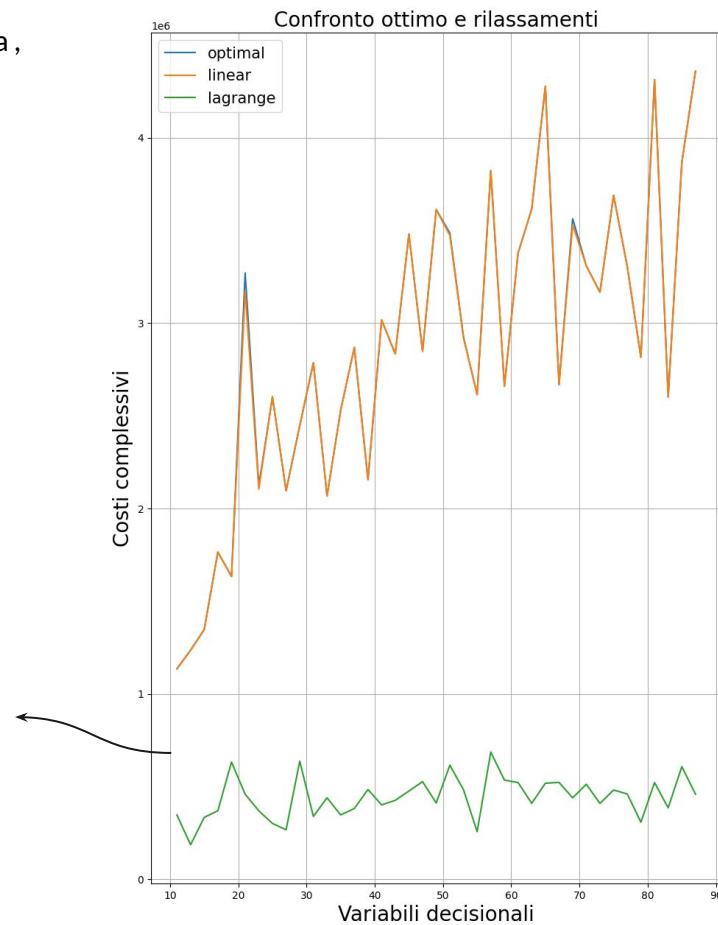
# UFL: dimensione fissa



# UFL: variando la dimensione (1/2)

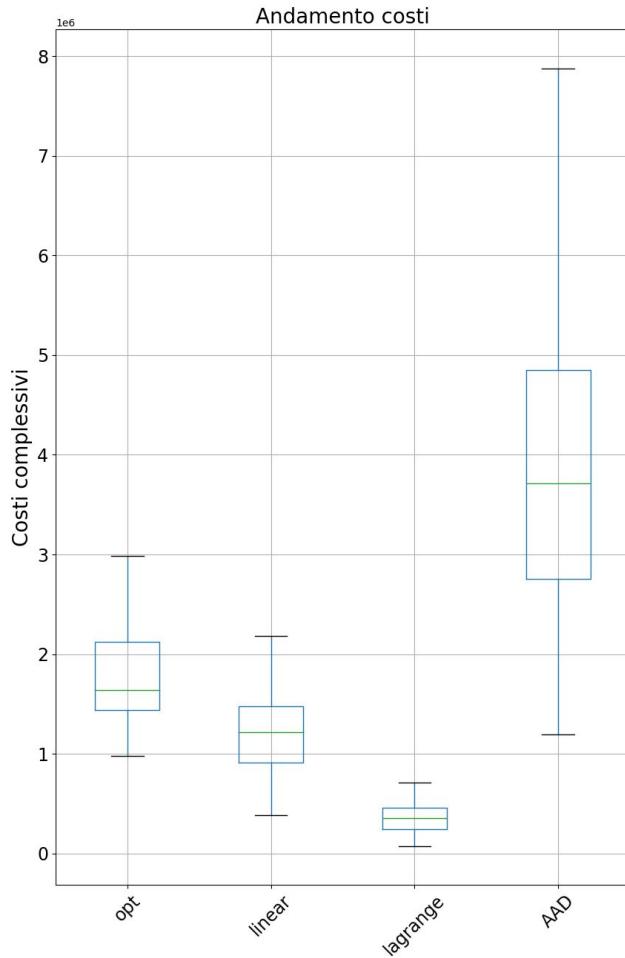


All'aumentare delle dimensioni del problema ,  
AAD degrada le prestazioni.

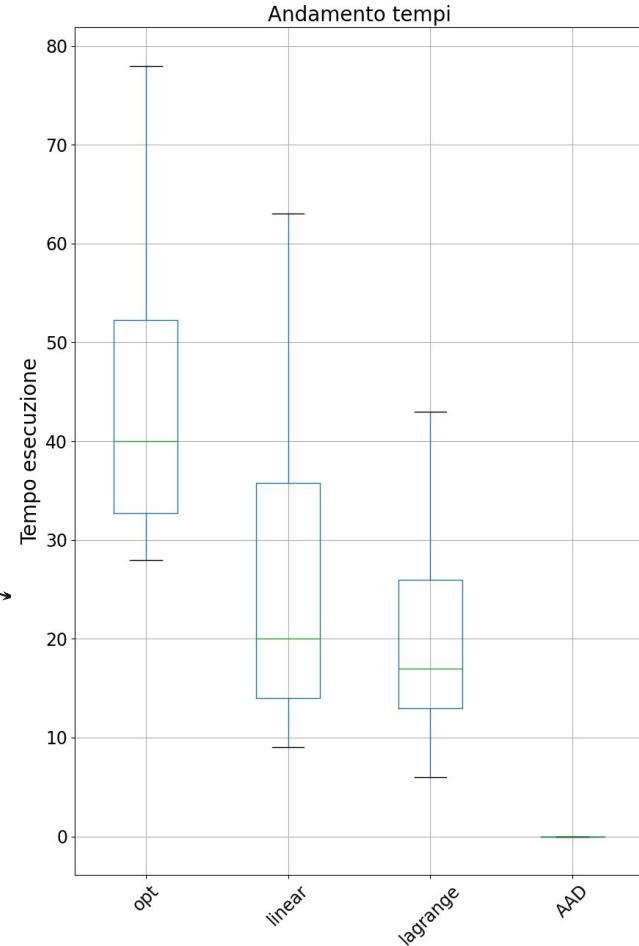


Il rilassamento lineare segue perfettamente la soluzione del PLI. Il rilassamento lagrangiano ha prestazioni peggiori.

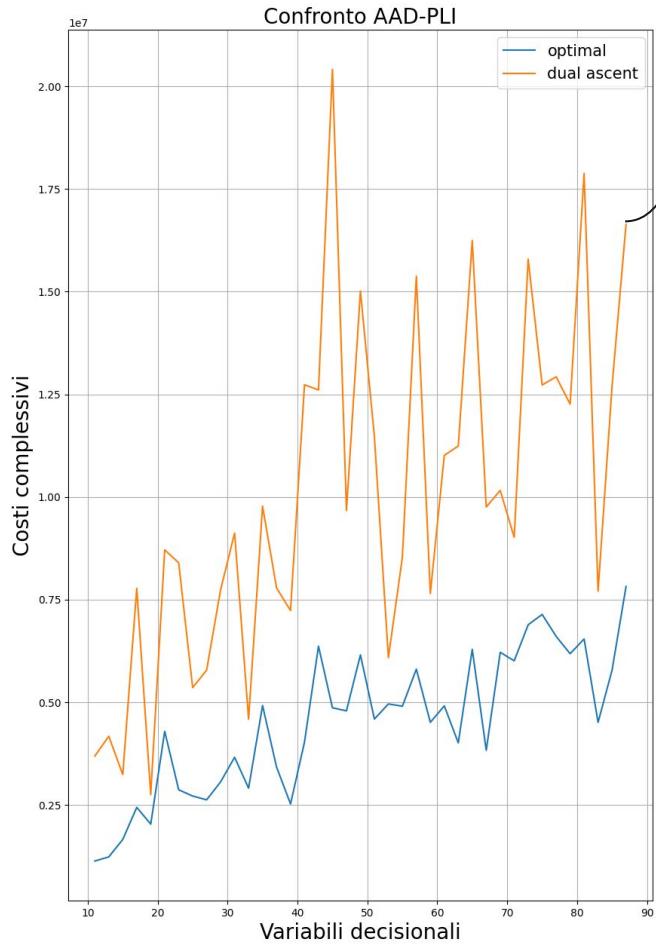
# CFL: dimensione fissa



Stessi risultati  
dell'analisi UFL

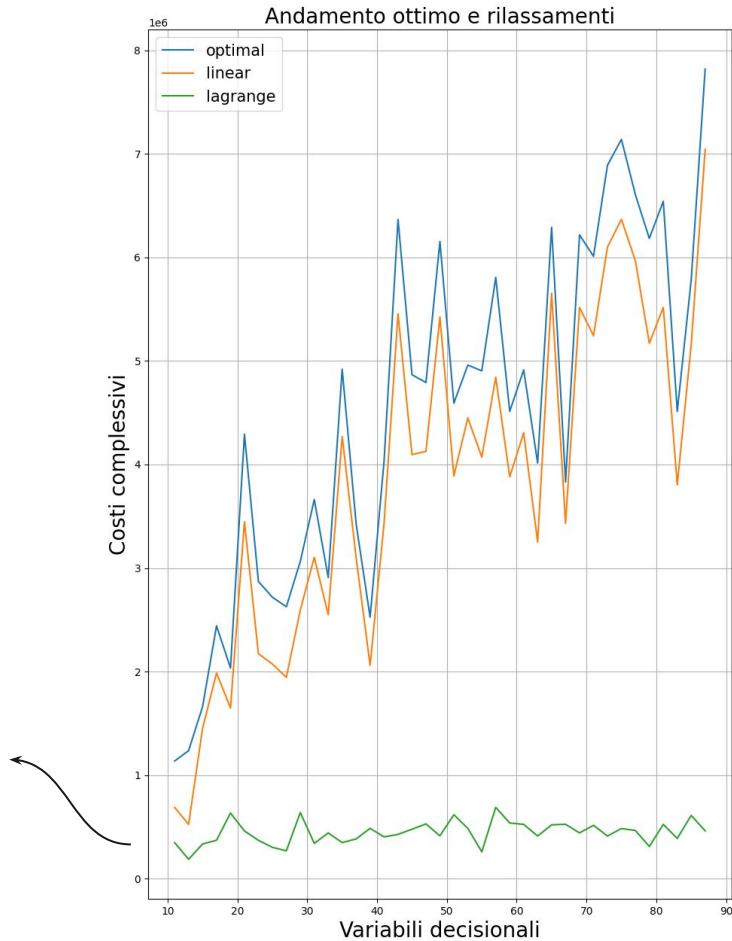


# CFL: variando la dimensione (1/2)



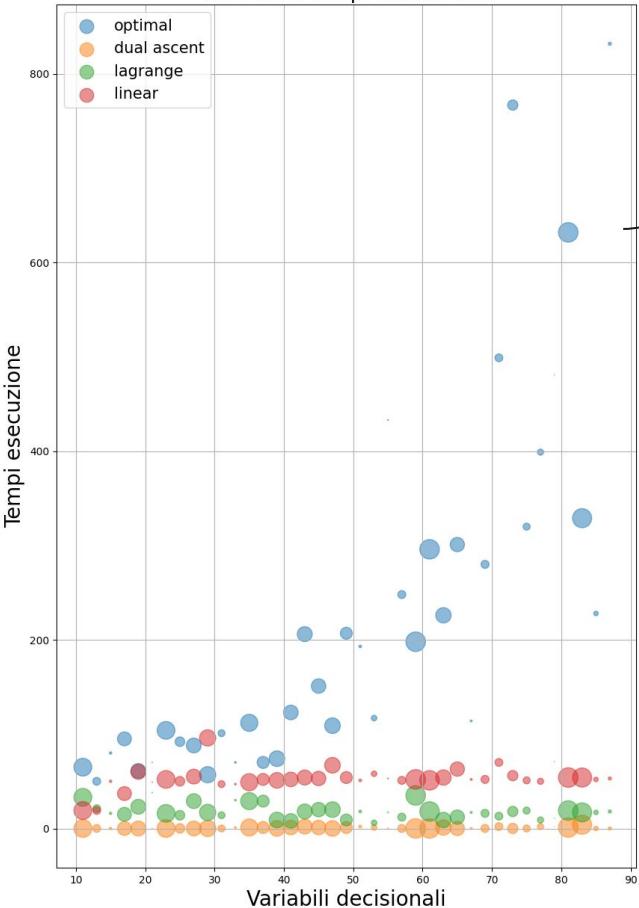
In questa versione del problema AAD segue bene la soluzione del PLI

Anche qui il rilassamento lineare è migliore di quello lagrangiano. Rispetto alla versione UFL il rilassamento lineare non segue perfettamente la soluzione del PLI.



# CFL: variando la dimensione (2/2)

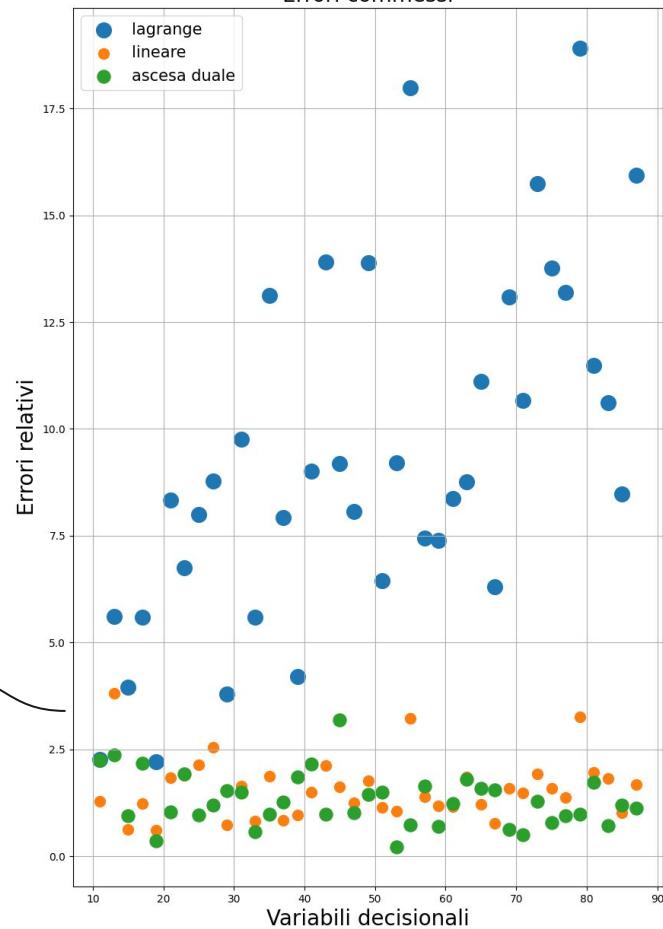
Confronto tempi esecuzione



Il tempo di esecuzione per trovare la soluzione del PLI cresce esponenzialmente.

Il rilassamento lagrangeano commette un errore in percentuale maggiore del rilassamento lineare.

Errori commessi



---

# Conclusioni

- Il rilassamento lineare permette di individuare un miglior lower bound rispetto al rilassamento lagrangiano, seppur più oneroso dal punto di vista computazionale.
- L'algoritmo di ascesa duale è poco oneroso dal punto di vista computazionale. Nella versione non capacitata non rappresenta una buona soluzione euristica. Nella versione capacitata l'algoritmo ha registrato prestazioni migliori.
- Si osserva che al crescere delle dimensioni del problema , il costo computazionale per trovare la soluzione ottima del PLI cresce in maniera esponenziale. Nel caso dei rilassamenti e con AAD non si osserva questo andamento