Optimization for Data Science

EXAMINATION PROJECT REPORT

Quadratic Minimum Cost Flow with Frank-Wolfe algorithm

Francesca Scotti (657524)
Valeria Picchianti (562913)

ACADEMIC YEAR 2022-2023

Indice

1 Introduzione

Sia P il problema di ottimizzazione

$$\min \{x^T Q x + qx : Ex = b, \;\; 0 \leq x \leq u\} \tag{1}$$

dove:

- $x,\ q \in \mathbb{R}^m$
- $Q \in \mathbb{R}^{mxm}$, matrice diagonale semidefinita positiva
- $E \in \mathbb{R}^{nxm}$, matrice di incidenza nodo–arco del grafo connesso con n nodi e m archi
- $b \in \mathbb{R}^n$, vettore dei bilanci dei nodi del grafo
- $u \in \mathbb{R}^m$, vettore a valori positivi che rappresenta la capacità superiore di ogni arco del grafo.

e sia A un algoritmo di ottimizzazione della classe dei metodi Frank-Wolfe (FW), basati sul conditional gradient (CG), utilizzato per risolvere problemi di ottimizzazione vincolati e convessi. Preso il problema P, questo report si propone di descrivere dettagliatamente un algoritmo di ottimizzazione A implementato per la risoluzione di P.

Il problema P è un problema di flusso di costo minimo, la cui funzione obiettivo è una funzione quadratica, convessa e separabile. La matrice diagonale Q ha valori non negativi che rappresentano i suoi autovalori; $Ex = b$ rappresenta in forma vettoriale i vincoli di conservazione del flusso. La matrice di incidenza E è costruita come segue:

$$E_{ij} = \begin{cases} 1 & \text{se l'arco } j \text{ è uscente dal nodo } i \\ -1 & \text{se l'arco } j \text{ è entrante nel nodo } i \\ 0 & \text{altrimenti} \end{cases} \tag{2}$$

Infine, x risulta limitata superiormente (u) e inferiormente (0): questo rappresenta un vincolo di capacità.

Di seguito si riportano i quattro step principali in cui è possibile schematizzare l'algoritmo risolutivo A; una discussione più dettagliata e approfondita verrà affrontata nella sezione 3, mentre le proprietà algoritmiche del problema verranno descritte in sezione 2.

0. Inizializzazione: viene definito il valore di x_0 appartenente alla regione ammissibile e si indica con k l'iterazione attuale.

1. Ricerca della direzione d: si determina d attraverso la minimizzazione della approssimazione lineare del problema data dallo sviluppo di Taylor del primo ordine della funzione di costo nell'intorno di x_k.

$$\bar{x} \leftarrow argmin\{\langle \nabla f(x), z \rangle : Ex = b, 0 \leq x \leq u\} \tag{3}$$

Risolvendo questo problema si ottiene $\bar{x}$ e la direzione è data da $d = \bar{x} - x_k$. [1]

2. Determinazione dello stepsize α (*line search*): può essere ottenuto come $\alpha = 2/(2+k)$ oppure trovando α che minimizzi $f(x_k + \alpha(d_k - x_k))$ rispettando il vincolo $0 \leq \alpha \leq 1$.

3. Aggiornamento posizione: $x_{k+1} = x_k + \alpha_k d_k$. Se $\langle \nabla f(x_{k+1}), d \rangle \geq -\varepsilon$ l'algoritmo termina in quanto è stata trovata l'approssimazione di x_*, altrimenti si incrementa il numero di interazione, ponendo $k = k+1$ e si ripetono gli step ripartendo dal punto 1.

2 Algoritmo di Frank-Wolfe

2.1 Proprietà della funzione

L'algoritmo in esame richiede che la funzione di costo sia una funzione convessa, a valori reali e differenziabile; occorre inoltre che i vincoli definiscano un insieme compatto e convesso in uno spazio vettoriale. Si procede dunque, in questo capitolo, a discutere le suddette proprietà ed altre caratteristiche del problema, tra cui la convergenza e la complessità.

Per quanto riguarda lo studio della convessità, occorre ricordare che una funzione si definisce convessa quando la disuguaglianza

$$f(\alpha x + \beta y) < \alpha f(x) + \beta f(y) \tag{4}$$

risulta soddisfatta per ogni $x, y \in \mathbb{R}^n$ e tutti gli α e β con $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$. La funzione $x^T Q x + qx$ è quindi sicuramente convessa, dal momento che la combinazione lineare di due funzioni convesse è ancora una funzione convessa: la forma quadratica $x^T Q x$ è infatti convessa poichè la matrice Q è una matrice semidefinita positiva e anche la funzione qx è convessa in quanto lineare.

Per quanto riguarda la convessità dei vincoli, è possibile notare che ciascuno di essi definisce un iperpiano affine e la loro intersezione definise un sottospazio affine, che è ancora convesso. La conoscenza di queste informazioni risulterà particolarmente importante, poichè permetterà di dire che il minimo trovato con l'algoritmo sarà un minimo globale. La funzione è quadratica con Q semi definita positiva e risulta quindi differenziabile con derivata seconda non negativa nel dominio. Per poter successivamente discutere alcuni possibili risultati di convergenza dell'algoritmo Frank-Wolfe, è necessario notare che la funzione è L-smooth, ovvero è differenziabile e il suo gradiente è L-continuo con costante di Lipschitz L positiva[9]. Infatti, dal momento che la matrice Hessiana $H(x)$ delle derivate seconde è limitata superiormente (più grande autovalore), essendo uguale semplicemente a $2Q$ abbiamo che:

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \iff ||\nabla^2 f(x)|| \leq L \tag{5}$$

[1] Dal momento che i vincoli che definiscono la regione ammissibile sono vincoli lineari, il sotto problema allo step 1 è un problema di programmazione lineare.

quindi

$$L = 2||Q|| = 2\lambda_{max}. \tag{6}$$

Sapendo che Q è semidefinita positiva, L sarà proporzionale all'autovalore più grande di Q, λ_{max}. Intuitivamente, questo ha senso dal momento che l'autovalore più grande misura la "curvatura" o "ripidità" della funzione $f(x)$. Se Q ha un autovalore più piccolo, allora la funzione varia lentamente ed è più 'smooth', mentre se Q ha un autovalore più grande, allora la funzione varierà più rapidamente.

La funzione è L-continua soltanto localmente, sulla regione ammissibile compatta, come afferma il seguente teorema [1]:
Sia f una funzione a valori reali definita su un insieme aperto e convesso Ω in $\mathbb{R}^n$, se f è convessa e $K \subset \Omega$ è un insieme compatto, allora la funzione f è Lipschitz continua su K.

Descritte le proprietà della funzione, nella sezione successiva verranno analizzate alcune proprietà dell'algoritmo.

2.2 Proprietà dell'algoritmo

2.2.1 Risultati di convergenza

A questo punto, procedendo con l'analisi della convergenza attesa e della complessità dell'algoritmo è utile citare il seguente teorema: [2].

Teorema 1: *Sia $f : X \to \mathbb{R}$ una funzione convessa e L-smooth su un set X compatto con diametro D, l'algoritmo Frank-Wolfe converge nel modo seguente:*

$$f(x) - f(x^*) \leq \frac{2LD^2}{k+3} \qquad for\ k \geq 1 \tag{7}$$

dove L è la costante di Lipschitz identificata in 2.1 – $L = \lambda_{max}$ di Q – , mentre D è il diametro dell'insieme X, definito come $D := \sup_{x,y \in X} ||x - y||$ e nel nostro caso $||x - y|| \leq ||u|| = D$.
La stima delle costanti di convergenza permetterà nelle sezioni successive di verificare sperimentalmente che i risultati rispettino le previsioni teoriche. Ci si aspetta pertanto che nel caso in oggetto l'algoritmo FW abbia un tasso di convergenza di $O(1/k)$, con k numero di iterazioni: l'algoritmo converge a un tasso sublineare, ovvero più lentamente rispetto ai tassi di convergenza superlineari o quadratici di altri algoritmi di ottimizzazione, come ad esempio il metodo di Newton o la discesa del gradiente coniugato. Tuttavia, il costo per iterazione dell'algoritmo di FW è relativamente basso rispetto ad altri algoritmi, poiché richiede solo di risolvere un problema di ottimizzazione lineare ad ogni iterazione.

Nel caso in cui f sia fortemente convessa con modulo τ, ovvero:

$$f(x) \text{ è } \tau\text{-convessa} \iff f(x) - \frac{\tau}{2}||x||^2 \text{ è convessa} \iff \nabla^2 f(x) \succeq \tau I \succ 0 \tag{8}$$

e quando valga l'assunzione che $x^* \in Int(X)$, allora i risultati di convergenza migliorano [2] [4], mostrando un tasso di convergenza lineare anzichè sublineare. In particolare vale il seguente teorema:

Teorema 2:[2] Sia $\mathcal{X}$ un insieme convesso compatto con diametro D e f una funzione L-smooth e dominata dal gradiente con costante c (c-gradient dominated). Si supponga inoltre che esista un punto di minimo $x^* \in \text{Int}(\mathcal{X})$, ovvero che esista $r > 0$ tale che $B(x^*, r) \subseteq \mathcal{X}$. Allora le iterazioni dell'algoritmo FW soddisfano:

$$f(x) - f(x^*) \leq \left(1 - \frac{r^2}{2Lc^2D^2}\right)^{t-1} \frac{LD^2}{2}$$

per ogni $t \geq 1$. In modo equivalente, il gap primale è al massimo $\varepsilon > 0$ dopo il seguente numero di ottimizzazioni lineari e calcoli del gradiente:

$$1 + \frac{2Lc^2D^2}{r^2} \ln\left(\frac{LD^2}{2\varepsilon}\right).$$

In particolare, se f è fortemente convessa con costante μ (e quindi $c = \frac{1}{2\mu}$-gradient dominated), allora per ottenere un gap primale inferiore o uguale a $\varepsilon > 0$, è necessario al massimo il seguente numero di ottimizzazioni lineari e calcoli del gradiente:

$$1 + \frac{LD^2}{\mu r^2} \ln\left(\frac{LD^2}{2\varepsilon}\right).$$

Un modo per rendere l'algoritmo tradizionale più efficiente è quello di prendere in considerazione che la differenza tra la funzione $f(x)$ e il suo first order model L_x può essere significativamente grande quando ci si allontani dal punto x in cui L_x viene calcolato. Per ovviare a questo problema è possibile restringersi ad una regione in cui il modello è effettivamente una buona approssimazione della funzione, detta *trust region*: ognuna delle variabili, rispetto alle iterazioni precedenti, potrà variare in modo limitato. Il problema che si presenta a questo punto è quello di scegliere una trust region box, che non sia nè eccessivamente grande, ma neanche troppo piccola dal momento che il numero delle iterazioni aumenterebbe. Si definisce dunque il box constraint:

$$\|z - x\|_\infty \leq \tau \tag{9}$$

e ci si propone di trovare un τ efficace, che renda la convergenza più veloce, sia questo costante oppure variabile ad ogni iterazione.

Esistono tuttavia anche delle varianti del tradizionale FW che si propongono di migliorare il tasso di convergenza dell'algoritmo, tra queste: [2]

- FW con Away Step;
- FW e Active Sets; [5]
- FW con Second Order Model anzichè First Order Model. [6]

Nelle sezioni successive viene analizzata la prima variante, Frank-Wolfe con Away Step (AFW), di cui riportiamo dei risultati di convergenza, che mostrano come questa abbia un tasso lineare in caso di τ-convessità.

Teorema 3: *Sia $P \subset \mathbb{R}^n$ un politopo* [2] *e f una funzione L-smooth e τ-convessa su P. Il tasso di convergenza dell'algoritmo AFW, per ogni $k \geq 1$, può essere scritto come:*

$$f(x_t) - f(x^*) \leq \left(1 - \frac{\tau\gamma^2}{4LD^2}\right)^{\left\lceil \frac{k-1}{2} \right\rceil} \frac{LD^2}{2} \tag{10}$$

dove D e γ sono rispettivamente il diametro e la pyramidal width del politopo. Quando la funzione non è τ-convessa ma solo smooth, vale il risultato all'equazione 7.

2.2.2 Criteri di terminazione

Per quanto riguarda i criteri di arresto di FW, è possibile trarre vantaggio dal fatto che la funzione $f(x)$ è convessa: anzichè usare come stopping criterion $\langle \nabla f(x), z \rangle \leq \epsilon$, in questo algoritmo la convessità permette di poter stimare realmente la distanza tra il valore ottimo della funzione f_* e il valore di f nel punto corrente. Infatti, dato che $f(x)$ è convessa, l'ottimo del first order model v_* costituisce un lower bound per il valore ottimo f_* . L'arresto dell'algoritmo avverrà pertanto quando la differenza $f(x) - v_*$ sarà sufficientemente piccola:

$$f(x) - v_* \leq \epsilon \tag{11}$$

ricordando che il valor ottimo del first order model che costituisce il limite inferiore all'ottimo della funzione è espresso come

$$v_* = f(x) + \langle \nabla f(x), \bar{x} - x \rangle \tag{12}$$

Dire che lo stop avviene quando $\langle \nabla f(x), \bar{x} - x \rangle = 0$ significa altresì dire che la direzione $d = \bar{x} - x$ è ortogonale al gradiente.

3 Implementazione dell'algoritmo

In questa sezione viene riportato lo pseudocodice dell'algoritmo implementato in Python, sia questo quello tradizionale (3.1), che la variante Away Step (3.2).

Nota sul Gap Anziché utilizzare un criterio di arresto basato esclusivamente sul gap tra il prodotto scalare del gradiente e la direzione corrente, abbiamo adottato una strategia più adattiva, implementando il gap relativo. Questo nuovo approccio calcola un limite inferiore basato sulla funzione obiettivo e sul gradiente nel punto successivo, confrontandolo con il miglior limite inferiore ottenuto finora. Il gap relativo, espresso come rapporto tra la

[2] La nostra regione ammissibile X verifica l'assunzione in quanto intersezione finita di semispazi chiusi e convessi.

differenza tra il valore attuale della funzione obiettivo e il miglior limite inferiore e il massimo tra il valore assoluto della funzione obiettivo attuale e 1, rappresenta il criterio di arresto. Se il gap relativo è inferiore a una soglia ϵ, l'algoritmo si interrompe, segnalando uno stato 'ottimale'.

3.1 Frank-Wolfe tradizionale

Come si può notare dallo pseudocodice, l'algoritmo inizia con l'inizializzazione di alcune variabili, tra cui x_k, che viene generata rispettando il vincolo $0 \leq x_k \leq u$. Dopodichè, iterativamente, viene risolto il sottoproblema di ottimizzazione lineare, usando il solver pyMCF-Simplex, che permette di individuare $\overline{x}$ e dunque la direzione di ricerca d_k. Il valore di α viene calcolato in modo diverso a seconda che si intenda procedere al calcolo in modo approssimato ($\alpha = \frac{2}{2+k}$) oppure trovando α che minimizzi $f(x_k + \alpha(d_k - x_k))$ (line search). L'individuazione di α consente poi di aggiornare la posizione x_{k+1}, il gradiente $\nabla f(x_{k+1})$ e il prodotto scalare tra quest'ultimo e la direzione di ricerca. L'algoritmo termina quando uno dei due criteri di stop seguenti risulta verificato: è stato raggiunto il numero massimo di iterazioni oppure il gap ha raggiunto il valore di epsilon prefissato.

Algorithm 1 Frank-Wolfe solving algorithm

procedure FW(Q, q, u, *epsilon*, *tau*, number of edges, max iter, file_path to generated instances, alpha_ottimo)
    $k \leftarrow 0$
    $\alpha \leftarrow 1$
    Initialize x_k by generating a random integer in $[0, u]$
    while $k < max\_iter$ **do** ▷ stopping criteria
        $gradient_k \leftarrow (2 \cdot Q \cdot \ x_k) + q$
        Create an instance of MCFSimplex using the gradient as linear cost vector
        Solve the problem using MCFSimplex
        if $solution\_status \neq 0$ **then**
            "Problem unfeasible!"
        else
            Get *optimal_solution* from MCF
            Get $\overline{x}$ from MCF intern
            $d_k \leftarrow \overline{x} - x_k$ ▷ search direction
            if *alpha_ottimo* **then**
                Call function $find\_optimal\_alpha(x_k, d, Q, q)$
            else
                $\alpha \leftarrow \frac{2}{2+k}$
            $x_{k+1} \leftarrow x_k + \alpha \cdot d_k$ ▷ compute new position
            $gradient_{k+1} \leftarrow (2 \cdot Q \cdot \ x_{k+1}) + q$ ▷ new gradient
            $scalar\_product \leftarrow \langle gradient_{k+1}, d_k \rangle$
            $lb = f(x_{t+1}) + \langle \nabla f(x_{t+1}), d \rangle$ ▷ Compute the lower bound
            if $lb \geq$ bestlb **then**
                bestlb $= lb$
            end if
            $relativeGap \leftarrow \frac{f(x_{k+1}) - bestlb}{\max(|f(x_{k+1})|, 1)}$ ▷ Compute the relative gap
            if $relativeGap \leq$ epsilon **then** ▷ Stopping criterion
                Set status to *optimal*
                break
            $x_k \leftarrow x_{k+1}$
            $k \leftarrow k + 1$
    end while
end procedure

3.2 Away Step

In questa sezione, presentiamo la variante dell'algoritmo di Frank-Wolfe nota come "Away-Step Frank-Wolfe (AFW)" con l'uso dell'away step, come proposta in [7] e [8]. Questo

algoritmo è stato sviluppato per affrontare problemi di ottimizzazione vincolata non lineare e si distingue per l'uso di un insieme attivo ("active set") per gestire i vincoli.

L'algoritmo inizia con l'inizializzazione dei punti e dei coefficienti associati agli elementi nell'insieme attivo. Successivamente, durante ciascuna iterazione, l'algoritmo decide tra due tipi di passi: lo step di FW tradizionale oppure quello di AFW. Questa scelta è guidata da criteri che considerano la direzione del gradiente e l'insieme attivo corrente. In particolare, quando il passo di Frank-Wolfe è scelto, l'insieme attivo viene aggiornato per includere un nuovo punto, mentre durante il passo di Away, l'insieme attivo viene modificato per rimuovere un punto.

Questo approccio con l'utilizzo dell'active set offre un meccanismo efficace per gestire i vincoli nell'ottimizzazione, consentendo all'algoritmo di adattarsi dinamicamente alle condizioni del problema e determinare i punti di minimo approssimato in modo efficiente.

Questa variante dell'algoritmo si propone come soluzione al fenomeno di "zig-zagging". Infatti, quando la soluzione ottimale si trova sul bordo della regione ammissibile, il tasso di convergenza delle iterazioni dell'algoritmo è piuttosto lento: $f(x(k)) - f(x^*) \leq \mathcal{O}\left(\frac{1}{k}\right)$(sublinear). Questo accade perché le iterazioni dell'algoritmo classico FW iniziano a zig-zagare tra i vertici che definiscono la faccia che contiene la soluzione x. In realtà, il tasso $1/k$ è stretto per una vasta classe di funzioni: Canon e Cullum [3] hanno dimostrato (approssimativamente) che $f(x(t)) - f(x^*) \geq \Omega\left(\frac{1}{k^{1+\gamma}}\right)$ per qualsiasi $\gamma > 0$ quando x^* giace su una faccia della regione ammissibile con alcune ulteriori ipotesi di regolarità.

Anche in questo caso si riporta lo pseudocodice.

Algorithm 2 Away-Step Frank–Wolfe (AFW)

Initialize x_0, $\mathcal{S}_0 \leftarrow \{x_0\}$ and $\lambda_{x_0,0} \leftarrow 1$
while $t <$ max_iter **do**
  $v_t^{\mathrm{FW}} \leftarrow \arg\min_{v \in \mathcal{P}} \langle \nabla f(x_t), v \rangle$
  $v_t^A \leftarrow \arg\max_{v \in \mathcal{S}_t} \langle \nabla f(x_t), v \rangle$
  if $\langle \nabla f(x_t), x_t - v_t^{\mathrm{FW}} \rangle \geq \langle \nabla f(x_t), v_{A,t} - x_t \rangle$ **then** ▷ Frank–Wolfe step
    $d_t \leftarrow x_t - v_t^{\mathrm{FW}}$
    $\gamma_{t,\max} \leftarrow 1$
  else ▷ Away step
    $d_t \leftarrow v_{A,t} - x_t$
    $\gamma_{t,\max} \leftarrow \frac{\lambda_{v_t^A,t}}{1-\lambda_{v_t^A,t}}$
  end if
  $\gamma_t \leftarrow \min\left(\frac{\langle \nabla f(x_t), d_t \rangle}{L\|d_t\|^2}, \gamma_{t,\max}\right)$
  $x_{t+1} \leftarrow x_t - \gamma_t d_t$
  if $\langle \nabla f(x_t), x_t - v_t^{\mathrm{FW}} \rangle \geq \langle \nabla f(x_t), v_t^A - x_t \rangle$ **then** ▷ Update coefficients and active set
    $\lambda_{v,t+1} \leftarrow (1+\gamma_t)\lambda_{v,t}$ for all $v \in \mathcal{S}_t \setminus \{v_t^A\}$
    $\lambda_{v_t^{\mathrm{FW}},t+1} \leftarrow \begin{cases} \gamma_t & \text{if } v_t^{\mathrm{FW}} \notin \mathcal{S}_t \\ (1-\gamma_t)\lambda_{v_t^{\mathrm{FW}},t} + \gamma_t & \text{if } v_t^{\mathrm{FW}} \in \mathcal{S}_t \end{cases}$
    $\mathcal{S}_{t+1} \leftarrow \begin{cases} \mathcal{S}_t \cup \{v_{\mathrm{FW},t}\} & \text{if } \gamma_t < 1 \\ \{v_{\mathrm{FW},t}\} & \text{if } \gamma_t = 1 \end{cases}$
  else ▷ Update coefficients and active set
    $\lambda_{v,t+1} \leftarrow (1+\gamma_t)\lambda_{v,t}$ for all $v \in \mathcal{S}_t \setminus \{v_{A,t}\}$
    $\lambda_{v_{A,t},t+1} \leftarrow (1+\gamma_t)\lambda_{v_{A,t},t} - \gamma_t$
    $\mathcal{S}_{t+1} \leftarrow \begin{cases} \mathcal{S}_t \setminus \{v_{A,t}\} & \text{if } \lambda_{v_{A,t},t+1} = 0 \\ \mathcal{S}_t & \text{if } \lambda_{v_{A,t},t+1} > 0 \end{cases}$
  end if
  $lb = f(x_{t+1}) + \langle \nabla f(x_{t+1}), d \rangle$ ▷ Compute the lower bound
  if $lb \geq$ bestlb **then**
    bestlb $= lb$
  end if
  $relativeGap \leftarrow \frac{f(x_{t+1}) - bestlb}{\max(|f(x_{t+1})|,1)}$ ▷ Compute the relative gap
  if $relativeGap \leq$ epsilon **then** ▷ Stopping criterion
    Set status to *optimal*
    break
  $t+ = 1$
end while

4 Descrizione dei dati usati negli esperimenti

In questa sezione si presenta una breve descrizione dei dati utilizzati negli esperimenti per testare l'algoritmo. Per generare delle istanze del problema di flusso di costo minimo che fossero significative è stato fatto riferimento qui. Nello specifico, vengono forniti problemi di progettazione di reti a costi quadratici convessi generati in modo casuale, che si avvalgono del generatore *netgen* per costruire l'istanza "base" (lineare), alla quale vengono successivamente aggiunti i costi quadratici.[3]
Nel nostro caso, dal file *.dmx* si ricavano i valori di q, u, b, le caratteristiche proprie del grafo, cioè m e n, rispettivamente il numero degli archi e il numero dei nodi, e si ricostruisce inoltre la matrice di incidenza nodo-arco E. L'estrazione dei dati dai file è avvenuta implementando delle funzioni customizzate.

Per quanto concerne i costi quadratici, si è deciso di generare casualmente ciascun elemento Q_{ij} della matrice Q all'interno di un intervallo centrato su q_{ij}/u_{ij}, ammettendo solo valori non negativi. Inoltre, si è scelto di modulare l'ampiezza di tale intervallo attraverso un parametro α, il che consente, a seconda del valore selezionato, di generare costi quadratici più o meno grandi.

In aggiunta, poiché il problema P riguarda matrici Q semidefinite positive, abbiamo optato per l'impiego di un altro parametro, ρ. Quest'ultimo determina la percentuale di archi in cui il costo quadratico viene casualmente impostato a zero all'interno della matrice appena creata. In tal modo, si definisce una funzione in grado di generare matrici di costi quadratici con variazioni dimensionali basate su α e una variazione nella percentuale di zeri basata su ρ. Tale strategia facilita inoltre lo studio della convergenza dell'algoritmo perchè permette di differenziare facilmente i casi in cui $\nabla^2 f(x) \succ 0$ oppure $\nabla^2 f(x) \succeq 0$.

5 Esperimenti

In questa sezione vengono presentati i risultati ottenuti negli esperimenti condotti per testare le performance degli algoritmi. I grafici riportati hanno infatti lo scopo di evidenziare alcuni aspetti specifici di tali algoritmi, come gli andamenti di convergenza, tempi di esecuzione e andamento dei gap. Si riportano i risultati per gli esperimenti condotti su istanze con 1000 nodi.

5.1 Scelta dello stepsize

L'algoritmo tradizionale di Frank-Wolfe prevede la possibilità di scegliere tra due famiglie di step size differenti: $\alpha = \frac{2}{2+k}$ oppure trovando α ottimo tramite line search, vale a dire cercare α che minimizzi $f(x_k + \alpha(d_k - x_k))$. L'obiettivo dei primi esperimenti è quello di stabilire se uno dei due permetta di avere performance significativamente migliori rispetto all'altro.

[3] `https://pypi.org/project/pynetgen/`

Dalla figura 1, emerge che l'utilizzo della line search per la scelta dello step size è la scelta migliore per quanto riguarda l'efficienza dell'algoritmo di Frank wolfe tradizionale. Infatti, si nota che la funzione obiettivo peggiora in alcune iterazioni dell'algoritmo che utilizza $\alpha = \frac{2}{2+k}$. I grafici di figura 1 sono stati generati mantenendo il parametro `epsilon = 1e-3` seppur abbastanza grande, poichè l'obiettivo principale era quello di osservare il comportamento dell'algoritmo anzichè il raggiungimento dell'ottimo (che viene comunque trovato da entrambe le versioni dell'algoritmo).

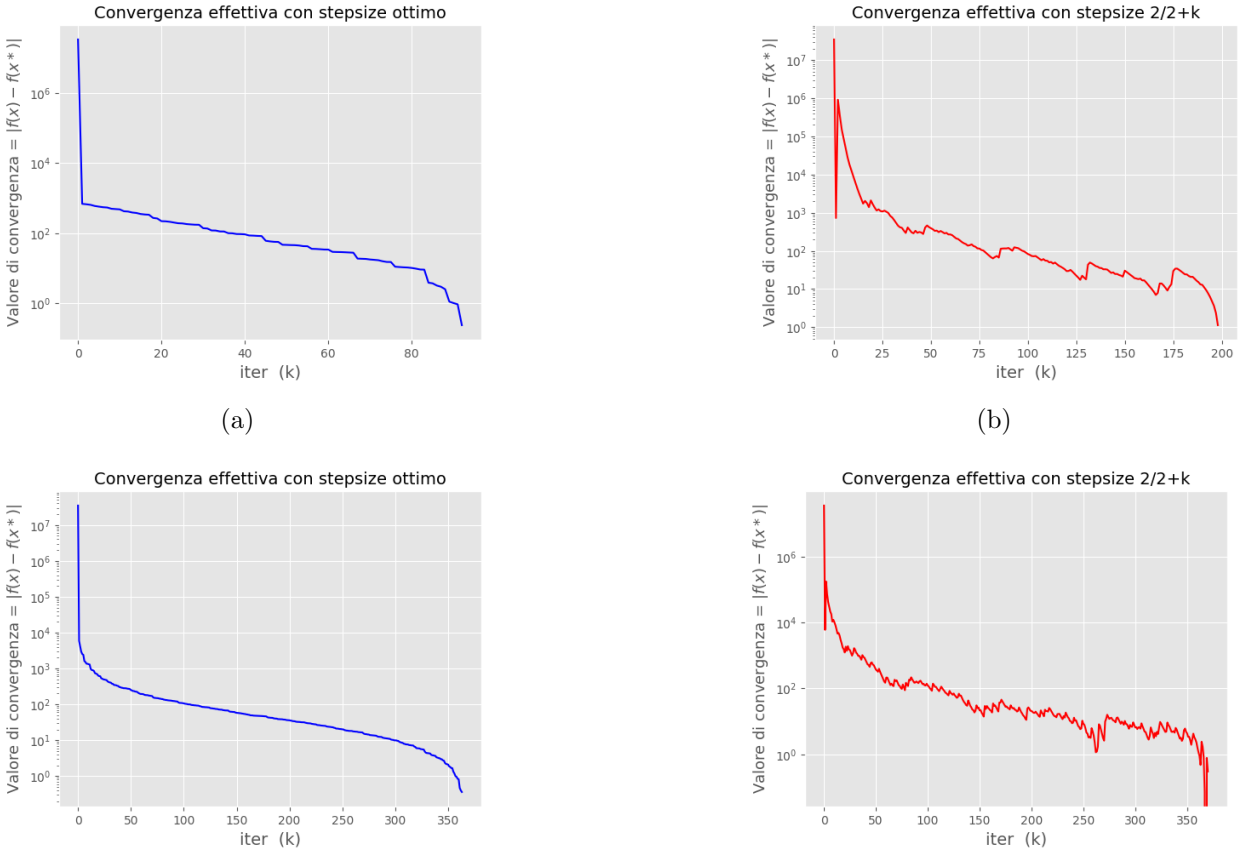
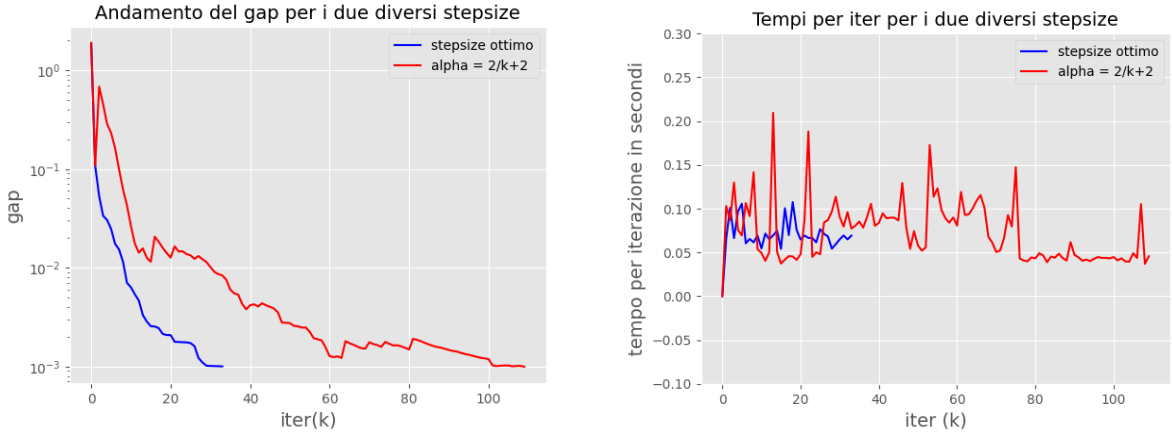


Figura 1: Convergenza effettiva con step size ottimo ed approssimato

I grafici in figura 1 mostrano la convergenza dell'algoritmo di FW tradizionale a seconda dello step size utilizzato. Ogni riga si riferisce alla soluzione di una istanza specifcia. I grafici a destra si riferiscono ad uno step size approssimato mentre quelli a sinistra allo step size ottimo. In Figura 2 è possibile notare la netta differenza dell'andamento del gap duale a seconda dello strategia adottata per lo stepsize e come variano i tempi in secondi per iterazione.

nome file dmx	**k optimal**	**k non_optimal**	**time optimal [s]**	**time non_optimal [s]**
netgen-1000-1-1-a-a-ns.dmx	20	162	0.7379	3.7583
netgen-1000-1-1-a-b-ns.dmx	47	961	1.5081	19.3207
netgen-1000-1-1-b-a-s.dmx	458	469	12.9778	9.4509
netgen-1000-1-1-b-b-s.dmx	22	184	0.6493	3.6977
netgen-1000-1-2-a-a-ns.dmx	144	1393	4.5109	31.5130
netgen-1000-1-2-a-a-s.dmx	70	352	2.3024	11.9585
netgen-1000-1-2-a-b-s.dmx	443	545	17.6035	14.2826
netgen-1000-1-2-b-a-ns.dmx	30	173	1.5183	5.7076
netgen-1000-1-2-b-a-s.dmx	39	651	2.61050	26.6440

Tabella 1: Esperimenti stepsize: la tabella mostra come lo step ottimo abbia una migliore performance sia in termini di iterazioni e tempi di esecuzione.
Parametri costanti: `Q(alpha = 1000, p = 0.5), epsilon = 1e-3, tau = 0`.



(a) (b)

Figura 2: Andamento del gap e tempi per iterazione con i due tipi di step size

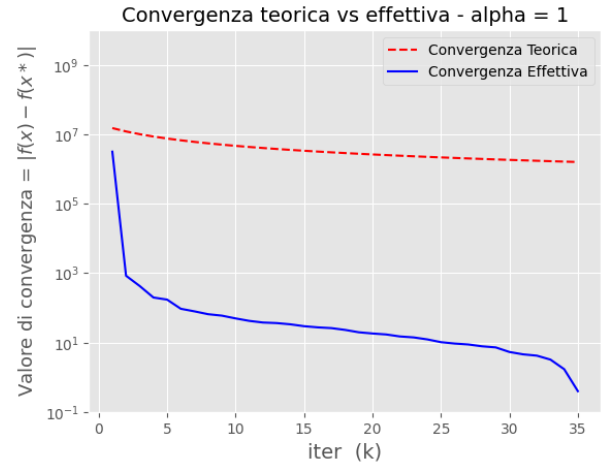
In figura 2, il grafico a) correttamente evidenzia come il gap duale (scala logaritmica) con stepsize ottimo (in blu) mostri un andamento monotono decrescente, mentre così non vale nel caso in cui si utilizzi lo stepsize $= \frac{2}{k+2}$ (Q semidefinita positiva, con percentuale di autovalori nulli `p = 70%, epsilon = 1e-3, tau = 0`). Il grafico b) intende invece mostrare come variano i tempi per iterazione in secondi nei due casi e nello specifico evidenzia che i tempi medi con $\alpha = \frac{2}{2+k}$ sono maggiori rispetto a α ottimo, come atteso. Infine, per escludere la possibilità che le migliori performance dell'algoritmo con α ottimo siano dovute semplicemente alla specifica istanza testata e graficata, si riporta una tabella che prende in considerazione dieci diversi file *.dmx*.

Tutti gli esperimenti successivi di cui riportiamo i risultati sono stati pertanto effettuati

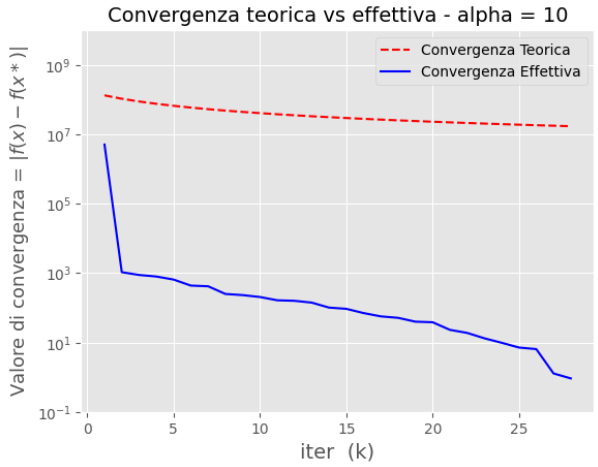
utilizzando lo step size con line search, abbandonando completamente $\alpha = \frac{2}{2+k}$.

5.2 Convergenza attesa ed effettiva di FW tradizionale

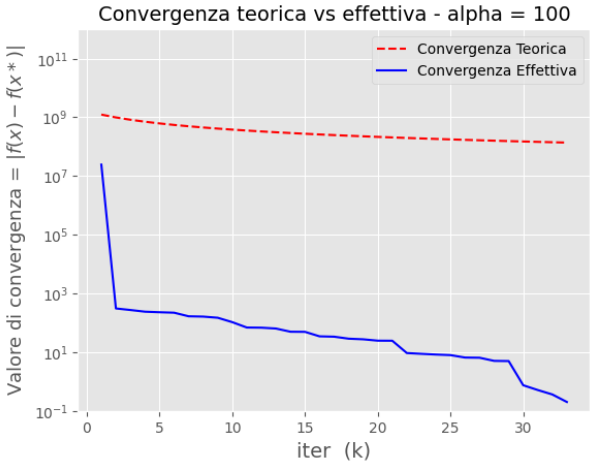
Nel caso dell'algoritmo FW tradizionale, come riportato in sezione 2, è stato possibile trarre vantaggio dalla formulazione esplicita dell'andamento dei valori di convergenza (con costanti tutte note) per fare un confronto tra la convergenza teorica attesa e quella effettiva. Dal momento che $f(x) - f(x^*)$ dipende anche dal più grande autovalore di Q, in Figura 3 si riportano i diversi andamenti al variare del range di valori assunti dalla norma degli autovalori della matrice Q (parametro alpha descritto in Sezione 4).



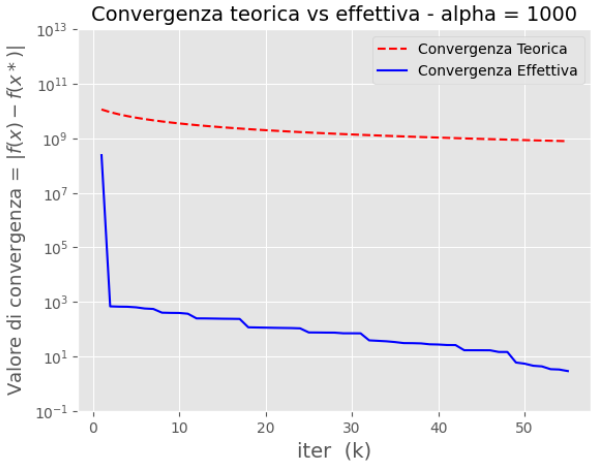
(a) alpha = 1



(b) alpha = 10



(c) alpha = 100



(d) alpha = 1000

Figura 3: Convergenza attesa ed effettiva

Nella fiugra 3 si osserva in blu come varia l'andamento di $f(x) - f(x^*)$ e, in rosso, l'andamento del valore di convergenza teorico (che è funzione di L, D, k) al variare di k (scala y logaritmica), per diversi valori del parametro *alpha*, che modella il range di valori assunti dalla norma degli autovalori della matrice Q, ovvero i costi quadratici del problema.

5.3 Effetto della Trust Region stabilization

Un altro aspetto che riteniamo interessante evidenziare è l'effetto della Trust region stabilization sulla convergenza dell'algoritmo. Nelle sezioni che seguono riportiamo i risultati ottenuti adottando due diverse strategie di implementazione.

5.3.1 Trust Region "ex ante"

Questa prima strategia di implementazione prevede di restringere la regione ammissibile in cui cercare la soluzione del problema di minimo lineare, quindi di fatto modificare lower bound e upper bound di X.

$$\overline{x} \leftarrow argmin\{\langle \nabla f(x), z \rangle : Ex = b, \quad lb_{\text{trust}} \leq x \leq ub_{\text{trust}}\} \tag{13}$$

con $\quad ub_{\text{trust}} = max(0, x - tau \cdot u)$ e $\quad lb_{\text{trust}} = min(u, x + tau \cdot u)$.

La strategia, seppur teoricamente corretta, ha condotto a delle inconsistenze rilevate durante gli esperimenti. Nello specifico, nella maggior parte degli esperimenti, il problema diviene unfeasible dopo qualche iterazione, qualora il valore di *tau* sia diverso da 0.5. Si ritiene che tali inconsistenze siano dovute a problemi di tipo numerico, poichè l'adozione della trust region stabilization impone sostanzialmente l'utilizzo di valori di tipo float, che necessitano controlli sull'accuratezza non necessari in caso di valori interi.

I parametri (*kEpsFlw, kEpsDfct*) che regolano l'accuratezza dei controlli numerici del solver MCF possono essere impostati dall'utente, in particolare si dovrebbe modificare kEpsFlw, impostandolo ad un valore più piccolo di 1e-14 ma purtroppo ciò non è possibile utilizzando il wrapper Python della libreria originale. [4]

Riportiamo comunque i risultati ottenuti utilizzando il solver Gurobi[5] attraverso la libreria di Python Gurobipy.

[4] Utilizzando la funzione MCFClass.SetPar() otteniamo l'errore '*NotImplementedError*: wrong number or type of arguments for overloaded function' seppur rispettando il tipo di input supportato dalla funzione.

[5] Sebbene Gurobi sia un solver generico e il problema richieda un solver specifico, abbiamo comunque deciso di utilizzarlo per proporre un risultato con la trust region ex-ante

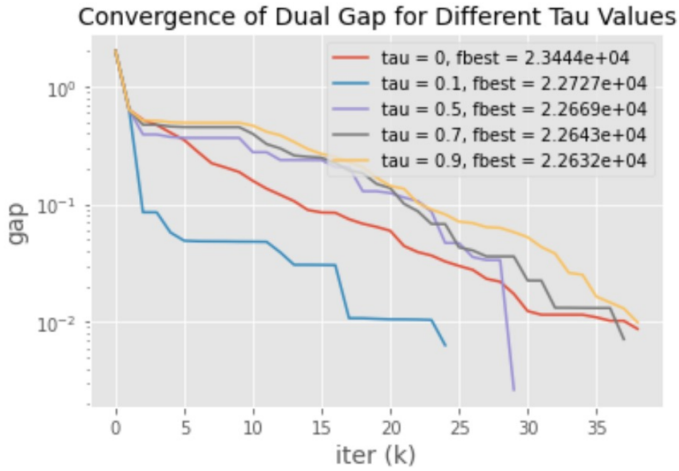


Figura 4: Convergenza Del Gap rispetto a vari valori di tau.

Il grafico in figura 4 mostra l'andamento del gap rispetto alle iterazioni per diversi valori di tau, speficati nella legenda; la soluzione ottima rimane simile variando il valore di tau, ma si nota un sostanziale miglioramento della convergenza per `tau = 0.1`. In questo caso è stato deciso di settare come stopping criterion `epsilon = 10e-2`.

Vista l'impossibilità di ottenere risultati significativi adottando questa strategia 'ex ante', abbiamo ritenuto opportuno investigare anche un'implementazione 'ex post' della trust regione, come descritto nel paragrafo che segue.

5.3.2 Trust Region "ex post"

Questa seconda strategia di implementazione agisce 'ex post', ovvero dopo aver calcolato la soluzione del problema lineare con MCFSolver, dunque obbligando v all'interno del box, come descritto dal seguente pseudo codice.

Algorithm 3 Update Trust Region and calculate d

if $\tau > 0$ **then**
  $ub_{\text{trust}} \leftarrow x + \tau \cdot u$
  $lb_{\text{trust}} \leftarrow x - \tau \cdot u$
  $\overline{x} \leftarrow \max(lb_{\text{trust}}, \min(\overline{x}, ub_{\text{trust}}))$
end if
$d \leftarrow \overline{x} - x$

Nell'algoritmo Frank-Wolfe, la direzione viene calcolata nel seguente modo: viene calcolato il gradiente e viene usato come vettore dei costi lineari nel problema da risolvere, ottenendo l'ottimo $\overline{x}$; se il parametro τ è maggiore di zero, si calcolano lb e ub come i limiti inferiore e superiore della trust region come mostrato nello pseudo codice. Si calcola $\overline{x}$ come

il massimo tra lb, il minimo tra $\overline{x}$ e ub. La direzione d_k viene calcolata come $\overline{x} - x_k$: viene considerata una direzione di discesa quando soddisfa la condizione $f(x + \alpha d) \leq f(x)$, dove f rappresenta la funzione obiettivo da minimizzare e α è una costante positiva che indica la lunghezza del passo lungo la direzione d.

Nel caso in cui il punto $\overline{x}$ venga limitato all'interno del range $[x - tau \cdot u, x + tau \cdot u]$, la direzione $d = \overline{x} - x$ rimane comunque una direzione di decrescita e la nuova soluzione $\overline{x}$ rispetta i vincoli definiti dal problema, risultando dunque ammissibile ($\nabla f(x)^T d < 0$).

Pertanto, costringere $\overline{x}$ all'interno dell'intervallo assicura che la direzione d continui a rappresentare una direzione di discesa per la funzione obiettivo, rispettando al contempo i vincoli del problema, come richiesto dall'algoritmo di Frank-Wolfe.

In tabella 2 riportiamo i risultati degli esperimenti ottenuti utilizzando questa implementazione di stabilizzazione. Come si può notare, nella maggior parte dei casi, il valore di tau che sembra condurre a riduzioni del numero di iterazioni e dunque di tempo, è $tau = 0.7$. Sebbene i valori 'ottimi' trovati nei diversi casi siano piuttosto simili tra loro, abbiamo deciso di testare la stessa istanza con gli stessi parametri (senza trust region) nel solver `cvx.py` ottenendo come valore ottimo 22466.20. Gli esperimenti e i confronti riportati nelle sezioni a seguire vengono effettuati mantenendo attiva la stabilizzazione con tau = 0.7.

tau	**k tot**	**best f value**	**tempo tot**
0.0	1241	22467.00	15.59
0.1	2000	21010.15	40.22
0.3	2000	22001.31	85.81
0.5	2000	22301.85	82.59
0.7	34	22438.22	0.86
0.9	820	22466.54	26.70

Tabella 2: Esperimenti per testare la trust region effettuati con diversi valori di tau, su un'istanza con Q fissa `(a = 100, p = 0.5), max_iter = 2000, epsilon = 1e-4.`

5.4 Effetto del parametro p - percentuale di autovalori nulli di Q

Nel corso di questa sezione, ci concentreremo sull'analisi dell'effetto del parametro p sui nostri risultati. Questo parametro svolge un ruolo fondamentale nella creazione della matrice Q, la quale contiene sulla diagonale i costi quadratici della funzione di costo. Il parametro p, come spiegato in sezione4, è responsabile di determinare la percentuale di zeri all'interno di questa matrice, introducendo casualmente zeri nei costi quadratici di alcuni archi. Questa variazione strategica ci permette di studiare le implicazioni sulla convergenza dell'algoritmo e di distinguere casi in cui la matrice Hessiana $\nabla^2 f(x)$ è definita positiva o semidefinita positiva. Nel seguente paragrafo, esamineremo in dettaglio l'effetto di questo parametro p e quindi della struttura della matrice Q e sulle dinamiche del nostro problema di ottimizzazione. In particolare, tramite il grafico di figura 5 osserviamo come il tempo di risuoluzione dell'algoritmo Frank Wolfe e quello dell'algoritmo Frank Wolf con away step cambi al variare del parametro p. Più il valore di p è piccolo, meno zeri sono contenuti nella matrice Q, e

dunque ci avviciniamo al caso $\nabla^2 f(x) \succ 0$. Al contrario, più il valore di p si avvicina ad uno, più ci si avvicina al caso in cui $\nabla^2 f(x) \succeq 0$.

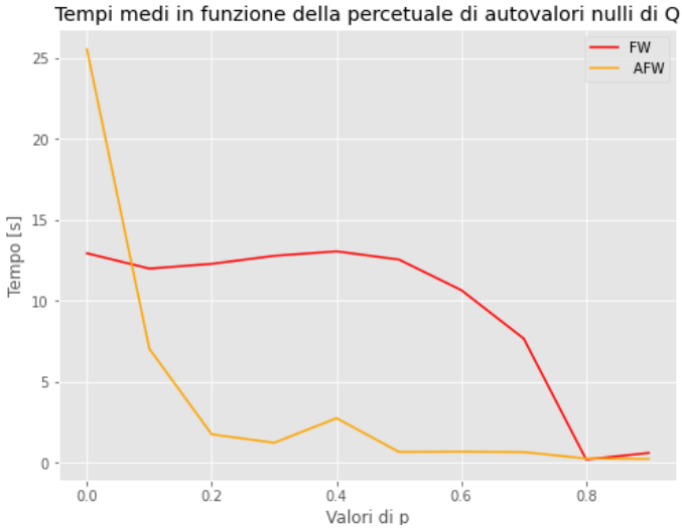


Figura 5: Tempi medi in funzione del parametro p.

Il grafico riporta l'andamento del tempo di esecuzione in secondi dei due algoritmi, AFW e FW, al variare del parametro p. Si osserva che il tempo impiegato da FW subisce piccole variazioni per valori di p fino 0.6, dopodichè inizia a diminuire; nel caso di AWF, la diminuzione del tempo è più netta.

5.5 Confronto tra FW tradizionale e Away-Step

Abbiamo deciso di condurre un'analisi approfondita per confrontare i due principali algoritmi presentati: il metodo di Frank-Wolfe standard [6] e il metodo di Frank-Wolfe con approccio "away step". Questo confronto ci aiuta ad osservare comportamenti simili e diversi e a trarre conclusioni sulla loro efficienza. Nel prossimo paragrafo, presenteremo una serie di grafici che evidenziano le differenze che emergono tra questi due approcci quando applicati agli stessi dati di input: per prima cosa utilizzando una matrice Q definita positiva e successivamente semidefinita positiva, inoltre abbiamo deciso di introdurre un'ulteriore dimensione da analizzare - il parametro *alpha* - che permette di scalare la dimensione dei costi quadratici.

5.5.1 Confronto sui tempi di esecuzione

Prima di effettuare gli esperimenti riguardanti il confronto tra l'algoritmo FW tradizionale e la variante che utilizza l'Away Step, prestiamo attenzione al tempo medio impiegato da ciascun algoritmo per iterazione, per riuscire successivamente a generare dei grafici significativi. Come si nota dalla tabella 3 ogni iterazione di AFW impiega circa il doppio del

[6] Tutti i confronti vengono effettuati considerando FW tradizionale con $tau = 0.7$ e stepsize ottimo.

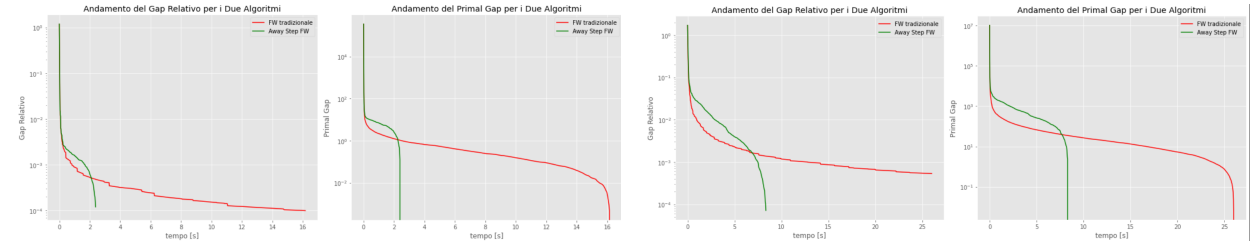
tempo dell'iterazione di FW. Per questo motivo i grafici non hanno lo scopo di confrontare l'andamento del gap per iterazione, quanto il suo andamento rispetto al tempo di esecuzione.

ID documento	FW (s)	AFW (s)	rapporto FW/AWF
1	0.0072	0.0152	2.111
2	0.0075	0.0163	2.173
3	0.0067	0.0181	2.701
4	0.0078	0.0155	1.987
5	0.0066	0.0182	2.701
media rapporto FW/AWF			2.334

Tabella 3: Tempo medio per iterazione per l'algoritmo di Frank wolfe (FW) e la variante che utilizza l'away step (AFW).

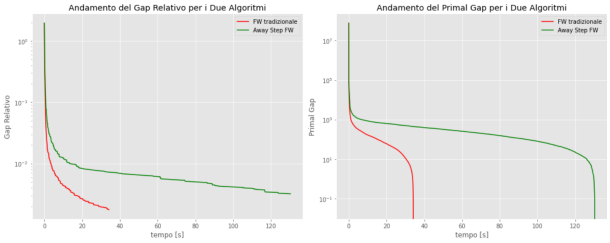
5.5.2 Caso 1 - Q definita positiva

Per la creazione dei primi grafici abbiamo deciso di generare una matrice Q definita positiva, utilizzando vari valori del parametro alpha. Abbiamo quindi utilizzato la stessa inizializzazione di x_0 ed inoltre abiamo deciso di mantenere costanti i parametri `epsilon = 1e-4`, `max_iter = 2000` in modo tale da poter trarre dagli esperimenti conclusioni significative. Ci troviamo nel caso in cui la funzione obiettivo è τ-convessa ed L-smooth, per cui ci aspettiamo un tasso lineare di convergenza anche per l'algoritmo AFW (Teorema 2).

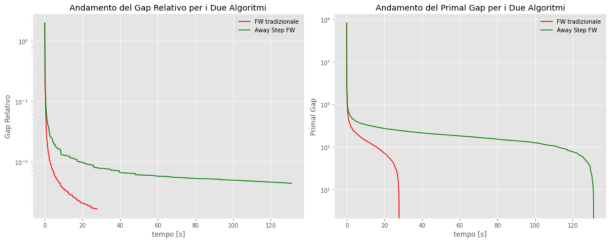


(a) alpha = 1

(b) alpha = 10



(c) alpha = 100



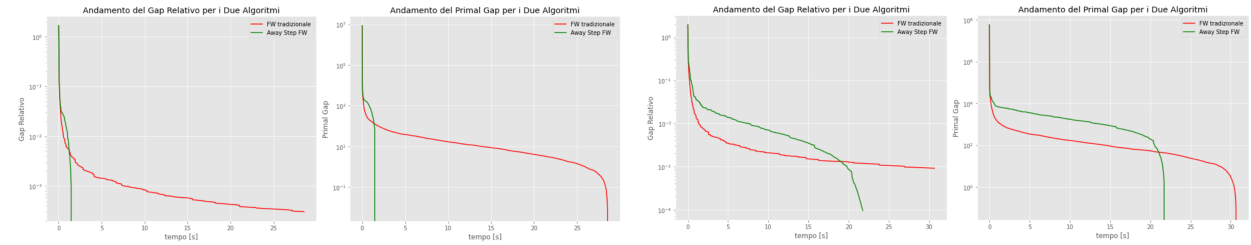
(d) alpha = 1000

Figura 6: Gap relativo e Primal gap in funzione del tempo, al variare di alpha, per i due algoritmi FW e AFW.

I grafici di figura 6 mostrano l'andamento del gap relativo e primale per gli algoritmi FW e AFW. Ogni coppia di grafici è distinta da un valore *alpha* utilizzato per la creazione della matrice Q. Si nota come all'aumentare di alpha il tempo di esecuzione aumenti. Inoltre, per alpha pari a 1 e 10 la convergenza dell'algotimo Away Step Frank Wolfe sembra essere migliore rispetto all'algoritmo tradizionale, mentre per alpha 100 e 1000, si preferisce la convergenza dell'algoritmo tradizionale, che non solo impiega meno tempo, ma raggiunge gap minori.

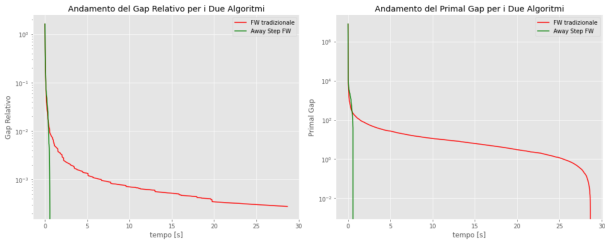
5.5.3 Caso 2 - Q semidefinita positiva

I grafici riportati in questa sezione si riferiscono ad istanze del nostro problema per le quali abbiamo deciso di generare una matrice Q definita positiva, utilizzando vari valori del parametro alpha. Abbiamo quindi, come nel caso precedente, utilizzato la stessa inizializzazione di x_0 e deciso di mantenere costanti i parametri `epsilon = 1e-4, max iter = 2000` in modo tale da poter trarre dagli esperimenti conclusioni significative. Ci troviamo nel caso in cui la funzione obiettivo è L-smooth ma non τ-convessa e per questo l'algoritmo tradizionale e la variante Away-Step hanno una convergenza attesa sublineare (Equazione 7).

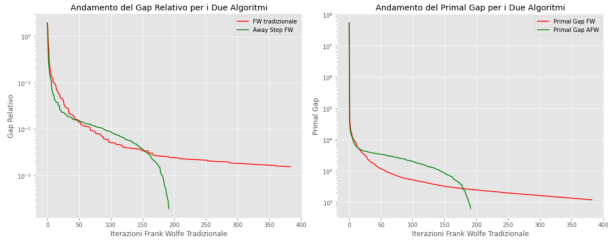


(a) p = 0.2 alpha = 10

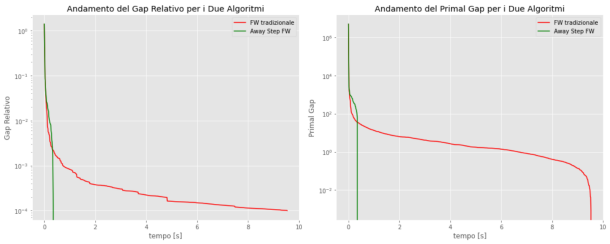
(b) p = 0.2 alpha = 100



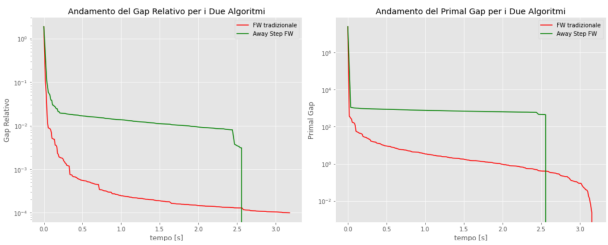
(c) p = 0.3 alpha = 10



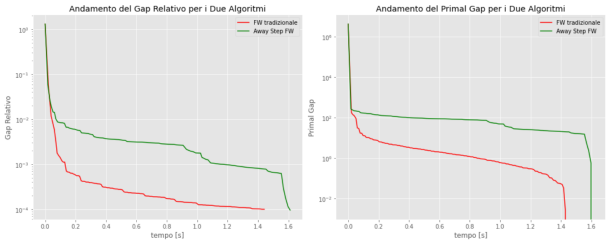
(d) p = 0.3 alpha = 100



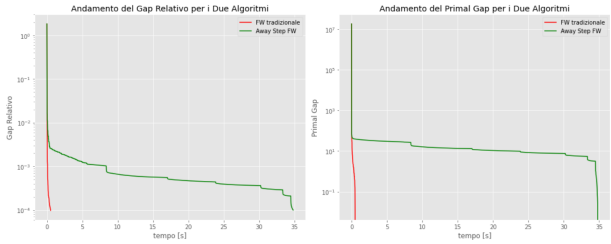
(e) p = 0.7 alpha = 10



(f) p = 0.7 alpha = 100



(g) p = 0.8 alpha = 10



(h) p = 0.8 alpha = 100

Figura 7: Gap relativo e Primal gap in funzione del tempo, al variare di alpha e p, per i due algoritmi FW e AFW.

I grafici di figura 7 mostrano l'andamento del dual gap per gli algoritmi FW e AFW. Ogni grafico è caratterizzato da specifici valori di p e *alpha*. Osservando i grafici si nota anche in questo caso come l'algoritmo Away Step abbia performance migliori dell'algoritmo tradizionale nel caso di alpha piccole, inoltre si nota che all'aumentare del valore di p ci sia un miglioramento delle performance dell'algoritmo tradizionale rispetto alla variante AFW.

6 Conclusioni

Abbiamo deciso di confrontare gli algoritmi da noi implementati con la soluzione del solver Gurobi. In particolare si confrontano i risultati di Frank Wolfe con tau = 0.7 (trust region ex-post), Frank Wolfe con Away Step e i risultati con il solver Gurobi, in cui è stato selezionato il metodo 'automatico' per la scelta dell'algoritmo di risoluzione (si nota che nella grande maggioranza dei casi viene utilizzato il Barrier method). Si evidenzia che i risultati sono ottenuti considerando la stessa Q con `alpha = 100, p= 0.6` per ogni riga della tabella 4, affinchè il confronto sia significativo. Inoltre vengono mantenuti costanti i parametri `epsilon=10e-4 max_iter=2500` per FW e AFW. Per quanto riguarda Gurobi, epsilon è settata 1e-4 di default.

ID documento	tempo FW	iterazioni FW	soluzione FW	tempo AWF	iterazioni AWF	soluzione AWF	tempo Gurobi	iterazioni Gurobi	soluzione Gurobi
1000-1-1-a-a-ns	15.131	1046	33450.150	25.190	762	32907.933	0.03	17	33447.861
1000-1-1-a-a-s	17.248	1193	11433.222	10.265	385	11254.812	0.05	18	11432.284
1000-1-1-a-b-ns	29.419	2000	14091.034	76.66	2000	14060.387	0.07	17	14089.934
1000-1-1-a-b-ns	21.484	1511	13115.98	33.678	864	12977.207	0.06	17	13115.102
1000-1-1-b-a-ns	29.319	2000	10781.901	7.205	280	10622.182	0.05	18	10780.711
1000-1-1-b-b-s	29.936	2000	49963.097	2.187	127	45136.709	0.05	19	49959.250
1000-1-2-a-a-ns	26.1993	2000	339094.560	17.240	602	336968.598	0.08	16	339044.742
1000-1-2-a-a-s	27.547	2000	44238.379	16.078	904	42573.961	0.05	16	44234.582
1000-1-2-a-b-ns	21.527	1606	20019.8089	6.969	351	19323.695	0.05	19	20018.521

Tabella 4: Confronto finale tra algoritmi

I risultati presentati nella tabella 4 mostrano un confronto tra le soluzioni ottenute attraverso i tre diversi approcci. Si osserva che, in termini di precisione della soluzione, le soluzioni ottenute tramite FW e AFW sono ragionevolmente vicine a quelle di Gurobi per la maggior parte dei casi considerati; tuttavia, FW e AFW richiedono un numero significativamente maggiore di iterazioni rispetto a Gurobi per convergere verso una soluzione accettabile.

In particolare, per problemi come '1000-1-1-a-a-ns' e '1000-1-1-a-b-ns', entrambe le varianti di Frank Wolfe ottengono soluzioni che differiscono in modo trascurabile dalla soluzione di Gurobi, pur impiegando un tempo e un numero di iterazioni molto superiori. D'altra parte, alcune istanze come '1000-1-1-b-b-s' mostrano una differenza più sostanziale tra le soluzioni di FW e AFW rispetto a Gurobi, con una precisione inferiore e un numero di iterazioni notevolmente più elevato.

In sintesi, la tabella 4 ci aiuta a capire che le soluzione trovate dagli algoritmi da noi implementati sono corrette e che sebbene l'algoritmo con Away Step sia spesso più veloce di Frank Wolf tradizionale, la soluzione ottenuta è meno precisa se confrontata con quella del solver Gurobi.

Riferimenti bibliografici

[1] Every convex function is locally lipschitz. *The American Mathematical Monthly*, 79(10):1121–1124, 1972.

[2] G. Braun, A. Carderera, C.W. Combettes, H. Hassani, A. Karbasi, A. Mokhtari, and S. Pokutta. Conditional gradient methods. `https://arxiv.org/pdf/2211.14103.pdf`, 2022.

[3] M. D. Canon and C. D. Cullum. A tight upper bound on the rate of convergence of frank-wolfe algorithm. *SIAM Journal on Control*, 6(4):509–516, 1968.

[4] Jacques GuéLat and Patrice Marcotte. Some comments on wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

[5] D. Zeffiro I.M. Bomze, F. Rinaldi. Active set complexity of the away-step frank-wolfe algorithm. `https://arxiv.org/abs/1912.11492`, 2019.

[6] S.J. Wright J. Nocedal. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, 2006.

[7] N. Soheili J. Pena, D. Rodrıguez. On the von neumann and frank-wolfe algorithms with away steps. `https://arxiv.org/pdf/1507.04073.pdf`, 2018.

[8] M. Jaggi S. Lacoste-Julien. On the global linear convergence of frank-wolfe optimization variants. `https://arxiv.org/pdf/1511.05932v1.pdf`.

[9] Xingyu Zhou. On the fenchel duality between strong convexity and lipschitz continuous gradient. 03 2018.