Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Magistrale in Ingegneria Informatica

Strategie di Ottimizzazione Globale per Problemi Quadratici Standard

Candidato
Mirco Ceccarelli

Relatori
Prof. Fabio Schoen
Prof.ssa Paola Cappanera

Correlatore
Dott. Matteo Lapucci

Anno Accademico 2021/2022

Abstract

I problemi di ottimizzazione quadratica standard (StQP) sono problemi quadratici con vincoli lineari semplici la cui risoluzione si concentra nella ricerca dei loro minimi globali. Gli StQP sono in generale non convessi ed appartengono alla classe dei problemi NP-difficili. In questo lavoro di tesi è stato applicato un algoritmo di decomposizione veloce, *Sequential Minimal Optimization (SMO)*, agli StQP con lo scopo di avvicinarsi il più possibile all'ottimo globale del problema. L'algoritmo SMO è stato utilizzato come risolutore locale all'interno di una strategia di ottimizzazione globale.
SMO aggiorna, ad ogni iterazione, due variabili scelte con una regola di selezione adeguata. Le principali caratteristiche dell'algoritmo sono che:

- Le due variabili vengono aggiornate risolvendo un sotto-problema che, sebbene non convesso, può essere risolto analiticamente;
- La regola di selezione adottata garantisce la convergenza verso punti stazionari del problema.

Un punto fondamentale dell'algoritmo SMO è la scelta del punto di partenza da cui far partire il processo di ottimizzazione. In questa tesi sono state provate varie strategie di inizializzazione ed eseguito vari esperimenti, ma quella che ha dato migliori risultati in termini di prestazioni computazionali, è stata quella relativa alla strategia multi-start basata sul grafo di convessità associato al problema di partenza (**SMO-CG**). Tali esperimenti hanno mostrato che SMO-CG è una valida alternativa agli algoritmi allo stato dell'arte per i problemi quadratici standard (StQP).

Abstract

Standard quadratic optimization problems (StQPs) are quadratic problems with simple linear constraints whose resolution focuses in the search for their global minimums. StQPs are generally not convex and belong to the class of NP-hard problems. In this thesis work, a fast decomposition algorithm, Sequential Minimal Optimization (SMO), was applied to the StQPs with the aim of approaching as much as possible to the global minimum of the problem. The SMO algorithm was used as a local solver within a global optimization strategy. SMO updates, to each iteration, two variables chosen with an adequate selection rule. The main characteristics of the algorithm are that:

- The two variables are updated by solving an sub-production which, although not convex, can be resolved analytically;
- The selection rule adopted guarantees the convergence towards stationary points of the problem.

A fundamental point of the SMO algorithm is the choice of the starting point from which to start the optimization process. Various initialization strategies have been proven in this thesis and various experiments, but the one that has given better results in terms of computational performance, was that relating to the multi-start strategy based on the graph of convexity associated with the starting problem (**SMO-CG**). These experiments have shown that SMO-CG is a valid alternative to the state of the art algorithm for standard quadratic problems (StQPs).

Indice

Capitolo 1

Introduzione

Il presente studio costituisce un lavoro di tesi per il corso di laurea magistrale in ingegneria Informatica presso l'Università degli studi di Firenze. La tesi è stata svolta presso il GOL (Global Optimization Laboratory) sotto la supervisione del professore Fabio Schoen e del dottore Matteo Lapucci. L'elaborato presenta la seguente struttura:

- **Capitolo 1**: si introducono i problemi di programmazione quadratica standard (StQP).
- **Capitolo 2**: si descrive l'algoritmo di decomposizione SMO applicato alla classe dei problemi StQP.
- **Capitolo 3**: si presenta la tecnica utilizzata per questo lavoro di tesi basata sul grafo di convessità associato al problema di ottimizzazione considerato (**SMO-CG**).
- **Capitolo 4**: si mostrano gli esperimenti eseguiti ed i risultati ottenuti su vari dataset di problemi StQP.

- **Capitolo 5**: contiene le conclusioni relative allo studio ed individua un sottoinsieme di possibili lavori futuri.

1.1 Problemi StQP

La programmazione quadratica (*Quadratic Programming (QP)*) è il processo di risoluzione di alcuni problemi di ottimizzazione matematica che coinvolgono funzioni quadratiche. In particolare si desidera ottimizzare (minimizzare o massimizzare) una funzione quadratica multivariata soggetta a vincoli lineari sulle variabili. La programmazione quadratica è quindi un tipo di programmazione non lineare [1].
I problemi di ottimizzazione che richiedono la programmazione quadratica sono spesso troppo complessi per essere risolti analiticamente e devono essere risolti numericamente.
La programmazione quadratica è ampiamente utilizzata in economia, finanza, ingegneria ed altri campi. Ad esempio, può essere utilizzata in finanza per il problema della "selezione del portafoglio" [2]. Con il termine portafoglio si intende un insieme di diverse attività. In particolare un portafoglio di titoli è dato dall'insieme di investimenti effettuati nei mercati finanziari. Si supponga che un portafoglio contenga n asset diversi. Il tasso di rendimento dell'asset i è una variabile casuale con valore atteso m_i. Il problema è quello di trovare quale frazione x_i deve essere investita in ogni asset i per minimizzare il rischio. Il problema inoltre è soggetto a un tasso di rendimento minimo specificato. Sia dunque C la matrice della covarianza del tasso di rendimento degli asset. Il modello classico per minimizzare il rischio del portafoglio è descritto da

$$\frac{1}{2}x^T Cx$$

ed è soggetto a un insieme di vincoli. Il rendimento atteso non dovrebbe essere inferiore al tasso minimo di rendimento del portafoglio r che l'investitore desidera:

$$\sum_{i=1}^{n} m_i x_i \geq r.$$

La somma delle frazioni di investimento x_i deve essere pari a uno:

$$\sum_{i=1}^{n} x_i = 1,$$

Essendo inoltre frazioni (o percentuali), devono essere numeri tra zero e uno:

$$0 \leq x_i \leq 1,\ i = 1, \ldots, n.$$

Poiché quindi la funzione obiettivo per minimizzare il rischio del portafoglio è quadratica con vincoli lineari, il problema di ottimizzazione risultante è un problema di programmazione quadratica (QP).

La programmazione quadratica è generalmente NP-difficile [3], il che significa che non esiste algoritmo noto che possa trovare una soluzione ottima in tempo polinomiale. Più formalmente, un problema H è NP-difficile se e solo se ogni problema NP (*Nondeterministic Polynomial-time*) L è polinomialmente riducibile ad H, ovvero tale che $L \leq_T H \quad \forall L \in NP$. La classe dei problemi NP-difficili ha una grande rilevanza sia teorica che pratica. In pratica, dimostrare che un problema di calcolo è equivalente a un problema notoriamente NP-difficile significa dimostrare che è praticamente impossibile trovare un modo efficiente di risolverlo [4].

In questo lavoro sono stati considerati gli *Standard Quadratic Optimization Problems* (StQPs), ovvero i problemi che sono nella seguente forma [5]:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & e^T x = 1 \\ & x \geq 0 \end{aligned} \tag{1.1}$$

Dove:

- $Q \in \mathbb{R}^{n \times n}$ è una matrice simmetrica indefinita;
- $c \in \mathbb{R}^n$;
- e è un vettore composto di tutti uno in $\mathbb{R}^n$.

L'insieme di tutti i vincoli lineari di uguaglianza e disuguaglianza è comunemente chiamato *simplesso standard n-dimensionale* e per semplicità di notazione viene indicato con Δ_n.
Gli StQPs sono in generale problemi NP-difficili [3] e sono risolvibili in tempo polinomiale solo in particolari casi, ovvero quando [6]:

- La matrice Hessiana Q è semi-definita positiva (caso convesso);
- La matrice Hessiana Q è semi-definita negativa (il minimo è raggiunto in uno degli n vertici del simplesso)

Bomze in [1] è stato il primo a vedere i problemi StQP come una preziosa classe di problemi, fornendo una descrizione completa delle proprietà teoriche e mostrando come alcuni problemi possano essere ricondotti al problema (1.1).
Bomze indica che i problemi come la "selezione del portafoglio" [7] vista precedentemente ed il trovare le *maximum clique* nei grafi [8] che sarà spiegata

in maniera più dettagliata nel capitolo 3, possono essere considerati come dei problemi StQP [9].
In ugual modo anche la genetica delle popolazioni, trattata da Kingman in [10], e la teoria evolutiva dei giochi, studiata da Bomze in [11], possono essere ricondotti agli StQP.
La struttura dei problemi StQP è simile alla formulazione duale dei problemi di training del *Support Vector Machine (SVM)* come verrà analizzato nel capitolo 2 [12].
Qualsiasi problema StQP può essere riformulato come un problema di programmazione lineare con un numero esponenziale di vincoli come è possibile vedere in [13]. Tale riformulazione è strettamente collegata alle condizioni KKT (condizioni di *Karush-Kuhn-Tucker*) che per i problemi StQP sono le seguenti [6]:

$$\begin{aligned} q_i x + c_i - \lambda &= \mu \quad i = 1, \ldots, n \\ \mu_i &\geq 0 \quad i = 1, \ldots, n \end{aligned} \tag{1.2}$$

dove:

- q_i è l'i-esima riga della matrice Q.
- λ è il moltiplicatore di Lagrange del vincolo $\sum_{i=1}^{n} x_i = 1$.
- μ_i con $i = 1, \ldots, n$ è il moltiplicatore di Lagrange del vincolo $x_i \geq 0$.

I problemi quadratici standard possono essere risolti in tempo finito da metodi di tipo *branch-and-bound* utilizzati per risolvere i più generali problemi di programmazione quadratica non convessa [14] [15] [16].
Questi approcci si basano su un'enumerazione implicita dei vincoli di complementarietà nelle condizioni KKT. I sotto-problemi risultanti sono dunque approssimati da rilassamenti semi-definiti o da rilassamenti semi-definiti poliedrici.

Con una semplice manipolazione delle condizioni KKT, un programma quadratico generale può essere formulato come programma lineare con vincoli di complementarietà (*Linear Program with Complementary Constraints (LPCC)*) e l'LPCC risultante può essere risolto da uno schema enumerativo come il branch-and-bound [17]. Con vincoli di complementarietà si intende dei vincoli in cui due variabili sono complementari tra loro.
In questo lavoro di tesi dunque è stato impiegato per risolvere tali problemi StQP un algoritmo di decomposizione veloce che può essere utilizzato come un efficiente risolutore locale all'interno di una strategia di ottimizzazione globale.

Capitolo 2

Algoritmo SMO per problemi StQP

Prima di analizzare l'algoritmo SMO per problemi StQP è necessario introdurre il problema di addestramento delle Support Vector Machines (SVM), la cui struttura come anticipato nel capitolo 1 può essere vista come un problema StQP.

È stato quindi proposto un algoritmo per risolvere SVM attraverso l'utilizzo di un metodo di decomposizione di tipo SMO, di cui verrà spiegato il funzionamento. L'algoritmo verrà dunque adattato per poter funzionare anche per problemi non convessi.

Le SVM sono dei modelli di apprendimento supervisionato associati ad algoritmi di apprendimento per la regressione e la classificazione. Nel caso di insiemi linearmente separabili, l'idea alla base delle SVM è di costruire un iperpiano di separazione (detto iperpiano ottimo) che massimizzi il margine di separazione degli elementi delle due classi. I loro punti di forza sono la velocità, il basso numero di iperparametri e l'efficacia con dataset di dimensioni relativamente piccola.

Le SVM sono state proposte per la prima volta in [18]. L'algoritmo SMO (Sequential Minimal Optimization) è stato introdotto in [19] e una dimostrazione della sua convergenza è stata data in [20].

2.1 Il problema di addestramento di Support Vector Machines

Il problema di addestramento di Support Vector Machines è riconducibile, mediante la teoria della dualità, ad un problema di programmazione quadratica convessa con vincoli lineari (l'insieme dei vincoli è costituito da un vincolo lineare e da vincoli di box) [19].
Considerando il problema della classificazione binaria con un insieme di dati (*dataset*) $(x_1, y_1), \ldots, (x_n, y_n)$, dove:

- x_i è un vettore in ingresso;
- $y_i \in \{-1, +1\}$ è la corrispondente etichetta binaria.

È necessario trovare l'iperpiano a massimo margine che divide il gruppo di punti x_i tra le due etichette, il quale è definito in modo tale che la distanza tra l'iperpiano e il punto più vicino x_i di entrambi i gruppi sia massimizzato. Qualsiasi iperpiano può essere riscritto come un insieme di punti x che soddisfano $w^T x - b = 0$, dove w è il vettore normale all'iperpiano, mentre $\frac{b}{\|w\|}$ determina l'offset dell'iperpiano dall'origine lungo il vettore normale w.

La formulazione primale del problema di addestramento (nel caso di classificazione) di una SVM è la seguente [12]:

$$\begin{aligned} \min_{w,b,\zeta} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{s.t.} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i \\ & \zeta_i \geq 0 \quad i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

Con $\zeta_i = \max(0, 1 - y_i(w^T x_i - b))$ $i = 1, \ldots, n$ dove n è il numero di dati di addestramento.

SVM separa quindi i vettori di addestramento in uno spazio mappato ϕ con un errore di costo C. In particolare l'iperparametro C determina l'influenza dell'errata classificazione sulla funzione obiettivo.

A causa della grande dimensione del vettore w, solitamente si risolve il problema (2.1) attraverso il suo problema duale Lagrangiano:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x - e^T x \\ \text{s.t.} \quad & y^T x = 0 \\ & 0 \leq x_i \leq C \quad i = 1, \ldots, n. \end{aligned} \tag{2.2}$$

Dove:

- $x \in \mathbb{R}^n$ è un vettore;
- Q è una matrice $n \times n$ simmetrica e semi-definita positiva;
- $e \in \mathbb{R}^n$ è il vettore i cui elementi sono tutti pari a uno.

In particolare il generico elemento Q_{ij} della matrice Q è dato da $Q_{ij} \equiv y_i y_j \phi(x_i)^T \phi(x_j)$, dove $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ è chiamata funzione kernel e ne esistono di vari tipi:

- Kernel polinomiale $K(x_i, x_j) = (x_i \cdot x_j + r)^d$;

- Kernel Gaussiano $K(x_i, x_j) = e^{\gamma\|x_i - x_j\|^2}$;
- Kernel Sigmoid $K(x_i, x_j) = \tanh(x_i \cdot x_j + r)$.

2.2 Sequential Minimal Optimization (SMO)

La matrice Hessiana Q in (2.2) è una matrice densa e diventa molto difficile memorizzarla quando si ha a che fare con problemi di dimensione elevata (ovvero quando si hanno molti dati di addestramento). Per questo motivo spesso la risoluzione di questi problemi viene affidata ad algoritmi di decomposizione che non richiedono la memorizzazione della matrice Q.
Ad ogni iterazione di un algoritmo di decomposizione viene selezionato un Working Set $W \subset \{1, \dots, n\}$, mentre $\overline{W} = \{1, \dots, n\} \backslash W$ indica il suo complementare. Il sotto-problema risultante quindi è il seguente:

$$\begin{aligned} \min_{x_W} \quad & f(x_W, x^k_{\overline{W}}) = \frac{1}{2} x_W^T Q_{WW} x_W - (e_W - Q_{W\overline{W}} x_{\overline{W}})^T x_W \\ \text{s.t.} \quad & y_W^T x_W = -y_{\overline{W}}^T x^k_{\overline{W}} \\ & 0 \leq x_W \leq C. \end{aligned} \tag{2.3}$$

Gli algoritmi di tipo "Sequential Minimal Optimization" (SMO) sono così chiamati perchè effettuano, ad ogni passo, l'ottimizzazione "minimale", ovvero rispetto a due sole variabili. Si ha quindi $|W| = 2$ e il sotto-problema (2.3) diventa

$$\begin{aligned} \min_{x_i, x_j} \quad & \frac{1}{2} \begin{bmatrix} x_i & x_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} - x_i - x_j + \begin{bmatrix} p_i & p_j \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} \\ \text{s.t.} \quad & y_i x_i + y_j x_j = b \\ & 0 \leq x_h \leq C \quad h = i, j \end{aligned} \tag{2.4}$$

con:

- $b = -\sum_{h \in \overline{W}} y_h x_h$;
- $p = Q_{W\overline{W}} x_{\overline{W}}$.

Uno dei vantaggi degli algoritmi SMO è che la soluzione a questo problema può essere facilmente essere trovata in modo analitico come si vedrà nella sezione 2.2.1.
Analizzando dunque la strategia di selezione ad ogni iterazione k delle variabili che costituiscono il working set, vogliamo che

$$x^{k+1} = (x_1^k, \ldots, x_i^{k+1}, \ldots, x_j^{k+1}, \ldots, x_n^k)$$

sia ammissibile e che

$$f(x^{k+1}) < f(x^k).$$

A questo scopo è necessario trovare, ad ogni iterazione, una direzione ammissibile di discesa con due e solo due componenti diverse da zero.

Proposizione 2.1 *L'insieme delle direzioni ammissibili per il problema (2.2) nel punto $\overline{x}$ è*

$$\mathcal{D}(\overline{x}) = \{d \in \mathbb{R}^n | y^T d = 0,\, d_i \geq 0 \forall i \in L(\overline{x}), d_i \leq 0 \forall i \in U(\overline{x})\} \tag{2.5}$$

dove

$$L(\overline{x}) = \{i \in \{1, \ldots, n\} | \overline{x}_i = 0\}$$

$$U(\overline{x}) = \{i \in \{1, \ldots, n\} | \overline{x}_i = C\}$$

Si vuole definire le direzioni in $\mathcal{D}(\overline{x})$ con due componenti diverse da zero. Dovendo risultare

$$y_i d_i + y_j d_j = 0$$

si pone

$$d_i = \frac{1}{y_i} \quad d_j = -\frac{1}{y_j}. \tag{2.6}$$

Inoltre:

- se $i \in L(\overline{x})$, ossia $\overline{x}_i = 0$, deve risultare $d_i \geq 0$, ovvero $y_i > 0$;
- se $i \in U(\overline{x})$, ossia $\overline{x}_i = C$, deve risultare $d_i \leq 0$, ovvero $y_i < 0$;
- se $j \in L(\overline{x})$, ossia $\overline{x}_j = 0$, deve risultare $d_j \geq 0$, ovvero $y_j < 0$;
- se $j \in U(\overline{x})$, ossia $\overline{x}_j = C$, deve risultare $d_j \geq 0$, ovvero $y_j > 0$.

È possibile notare che nel caso $0 < \overline{x}_i < C$ non ci sono vincoli sul segno di y_i e lo stesso si può dire per y_j. Si può quindi suddividere gli insiemi U e L nel modo seguente:

$$L(\overline{x}) = L^+(\overline{x}) \cup L^-(\overline{x}) \quad U(\overline{x}) = U^+(\overline{x}) \cup U^-(\overline{x})$$

dove:

$$L^+ = \{h \in L(\overline{x}) | y_h > 0\} \quad L^-(\overline{x}) = \{h \in L(\overline{x}) | y_h < 0\}$$

$$U^+ = \{h \in U(\overline{x}) | y_h > 0\} \quad U^-(\overline{x}) = \{h \in U(\overline{x}) | y_h < 0\}.$$

Infine si definisce gli insiemi

$$R(\overline{x}) = L^+(\overline{x}) \cup U^-(\overline{x}) \cup \{i | 0 < \overline{x}_i < C\} \tag{2.7}$$

e

$$S(\overline{x}) = L^-(\overline{x}) \cup U^+(\overline{x}) \cup \{i | 0 < \overline{x}_i < C\} \tag{2.8}$$

Proposizione 2.2 *La direzione $d^{i,j}$ definita in (2.6) è ammissibile in $\overline{x}$ per il problema (2.2) se e solo se $i \in R(\overline{x})$ e $j \in S(\overline{x})$.*

Proposizione 2.3 *La direzione $d^{i,j}$ definita in (2.6) è di discesa in $\overline{x}$ per il problema (2.2) se e solo se*

$$\frac{\nabla_i f(\overline{x})}{y_i} < \frac{\nabla_j f(\overline{x})}{y_j}. \tag{2.9}$$

Ora è possibile dunque definire lo schema generale di un algoritmo SMO:

Data: Sia $x^0 = 0$

1 $k = 0$;

2 $\nabla f(x^0) = -e$;

3 **while** *criterio di arresto non soddisfatto* **do**

4 Seleziona $i \in R(x^k)$, $j \in S(x^k)$ tali che $\frac{\nabla_i f(\overline{x})}{y_i} < \frac{\nabla_j f(\overline{x})}{y_j}$;

5 Poni $W = \{i, j\}$;

6 Risolvi analiticamente il problema (2.4);

7 Sia $x^\star_W$ la soluzione trovata al passo precedente;

8 Poni $x_h^{k+1} = \begin{cases} x_h^\star & \text{se } h \in W \\ x_h^k & \text{altrimenti} \end{cases}$;

9 Poni $\nabla f(x^{k+1}) = \nabla f(x^k) + Q_i(x_i^{k+1} - x_i^k) + Q_j(x_j^{k+1} - x_j^k)$;

10 $k = k + 1$;

11 **return** $x^\star = x^k$

Algoritmo 1: Algoritmo SMO generico

La regola di aggiornamento del gradiente deriva dal fatto che la funzione obiettivo è quadratica e, come si può notare dal passo 9, richiede l'utilizzo di sole due colonne della matrice Q (le colonne Q_i e Q_j), permettendo di risparmiare tempo e memoria. Da notare anche come la scelta del punto di partenza $x^0 = 0$ sia dovuta al fatto che in tale punto $\nabla f(x^0) = -e$, quindi non c'è bisogno di calcolare il gradiente nel punto iniziale.
Lo schema descritto genera una sequenza x^k tale che $f(x^{k+1}) < f(x^k)$.

Tuttavia tale condizione non è sufficiente per garantire la convergenza, vedremo quindi nella sezione 2.4.5 quali modifiche dovranno essere fatte per garantirla.

2.2.1 Calcolo della soluzione analitica per problemi a due variabili

In questa sezione sarà analizzato come poter risolvere il problema a due variabili. Si consideri dunque il problema (2.4) e l'insieme delle direzioni ammissibili (2.5).
Dato un punto ammissibile $\overline{x}$ e una direzione $d \in D(\overline{x})$, indichiamo con $\overline{\beta}$ il massimo passo ammissibile lungo la direzione d a partire da $\overline{x}$, ovvero:

- se $d_1 > 0$ e $d_2 > 0$, $\overline{\beta} = \min\{C - \overline{x}_1, C - \overline{x}_2\}$;
- se $d_1 < 0$ e $d_2 < 0$, $\overline{\beta} = \min\{\overline{x}_1, \overline{x}_2\}$;
- se $d_1 > 0$ e $d_2 < 0$, $\overline{\beta} = \min\{C - \overline{x}_1, C\}$;
- se $d_1 < 0$ e $d_2 > 0$, $\overline{\beta} = \min\{\overline{x}_1, C - \overline{x}_2\}$.

Sia inoltre $d^+ = \begin{pmatrix} \frac{1}{y_1} \\ -\frac{1}{y_2} \end{pmatrix}$.
Si mostra adesso la procedura di calcolo della soluzione analitica (PSA):

1 Se $\nabla f(\overline{x})^T d^+ = 0$ poni $x^\star = x$ e STOP;
2 Se $\nabla f(\overline{x})^T d^+ < 0$ poni $d^\star = d^+$;
3 Se $\nabla f(\overline{x})^T d^+ > 0$ poni $d^\star = -d^+$;
4 Se $\overline{\beta} = 0$ poni $x^\star = \overline{x}$ e STOP;
5 **if** $(d^\star)^T Q d^\star = 0$ **then**
6 $\quad \beta^\star = \overline{\beta}$;
7 **else**
8 $\quad \beta_{nv} = \frac{-\nabla f(\overline{x})^T d^\star}{(d^\star)^T Q d^\star}$;
9 $\quad \beta^\star = min(\overline{\beta}, \beta_{nv})$;
10 Calcola $x^\star = \overline{x} + \beta^\star d^\star$;

Algoritmo 2: Procedura di calcolo della soluzione analitica (PSA)

Proposizione 2.4 *La procedura PSA fornisce un vettore $x^\star$ che è soluzione ottima del problema (2.4).*

2.3 Un algoritmo per problemi di addestramento SVM non convessi

L'algoritmo SMO visto fin ora però non è capace di gestire il problema (2.2) se la matrice Q non è semi-definita positiva, ovvero se il problema è non convesso.

Infatti se la matrice Q non è semi-definita positiva, può capitare che una delle sotto-matrici a due variabili che vengono utilizzate nel corso della procedura, abbia determinante negativo. Ciò significa che la funzione quadratica del problema (2.4) è concava e l'utilizzo della procedura PSA (in particolare il passo 5) muove la soluzione verso il massimo. Non viene dunque rispettata

la proprietà richiesta dell'algoritmo di ottimizzazione di avere il valore della funzione obiettivo che diminuisce strettamente ad ogni iterazione. Inoltre potrebbe non essere ammissibile muoversi lungo una direzione positiva.
Tuttavia, come mostrato in [12], il caso non convesso può essere gestito con un piccolo cambiamento nel calcolo della soluzione analitica del problema a due variabili. L'intuizione è quella di trattare i casi in cui la matrice a due variabili ha determinante negativo allo stesso modo in cui vengono trattati i casi in cui il determinante è uguale a zero. Per fare ciò è sufficiente modificare il punto 5 della procedura PSA, che diventa quindi:

1 Se $\nabla f(\overline{x})^T d^+ = 0$ poni $x^\star = x$ e STOP;
2 Se $\nabla f(\overline{x})^T d^+ < 0$ poni $d^\star = d^+$;
3 Se $\nabla f(\overline{x})^T d^+ > 0$ poni $d^\star = -d^+$;
4 Se $\overline{\beta} = 0$ poni $x^\star = \overline{x}$ e STOP;
5 **if** $(d^\star)^T Q d^\star \leq 0$ **then**
6 $\quad \beta^\star = \overline{\beta}$;
7 **else**
8 $\quad \beta_{nv} = \frac{-\nabla f(\overline{x})^T d^\star}{(d^\star)^T Q d^\star}$;
9 $\quad \beta^\star = min(\overline{\beta}, \beta_{nv})$;
10 Calcola $x^\star = \overline{x} + \beta^\star d^\star$;

Algoritmo 3: Procedura di calcolo della soluzione analitica (PSA)

Quanto visto finora dunque è l'implementazione di un algoritmo per la risoluzione dei problemi di addestramento di SVM, che sia in grado di trovare una soluzione anche nel caso in cui il problema non sia non convesso.

2.4 Algoritmo SMO per gli StQP

La modifica di SMO analizzata nella sezione 2.3 è utile per i problemi StQP, poiché sono generalmente problemi non convessi e NP-difficili come visto nel capitolo 1.1.
Si riassume dunque quanto visto nella sezione 2.2 per poi presentare l'algoritmo SMO modificato per gli StQP.
In questo lavoro di tesi è stato dunque utilizzato un adattamento dell'algoritmo classico SMO affinché potesse essere applicato alla classe dei problemi StQP, garantendo sempre la proprietà di convergenza asintotica.
In generale negli schemi di decomposizione, ad ogni iterazione k, il vettore delle variabili x^k è diviso in due sotto-vettori $(x^k_W, x^k_{\overline{W}})$, dove [5]:

- l'insieme degli indici $W \subset \{1, \ldots, n\}$ identifica le variabili del sotto-problema che deve essere risolto (chiamato *working set*)
- $\overline{W} = \{1, \ldots, n\} \backslash W$

Quindi partendo da una soluzione corrente $x^k = (x^k_W, x^k_{\overline{W}})$, il sotto-vettore x^{k+1}_w è calcolato risolvendo il seguente sotto-problema:

$$\begin{aligned} \min_{x_W} \quad & f(x_W, x_{\overline{W}^k}) \\ \text{s.t.} \quad & e^T_W x_W = 1 - e^T_{\overline{W}} x^k_{\overline{W}} \\ & x_W \geq 0 \end{aligned} \tag{2.10}$$

Gli algoritmi Sequential Minimal Optimization (SMO) sono algoritmi di decomposizione il cui *working set* ha cardinalità esattamente uguale a due (che è la dimensione minima per questa classe di problemi).
Le principali caratteristiche dei metodi SMO sono le seguenti:

1. La soluzione del sotto-problema quadratico (2.10) può essere ottenuta in forma chiusa anche se il sotto-problema è non convesso.

2. Il gradiente della funzione obiettivo può essere aggiornato efficientemente usando solo due colonne della matrice Hessiana.

2.4.1 Definizione formale dell'algoritmo

La definizione formale dell'algoritmo *Sequential Minimal Optimization (SMO)* per la classe dei problemi StQP è la seguente:

Data: $x^0 \in \Delta_n$

Sia $k = 0$;

Sia $\nabla f(x^0) = Qx^0 + c$;

while *il criterio di arresto non è soddisfatto* **do**

  Seleziona il *working set* $W_k = \{i_k, j_k\}$ con una regola adatta;

  Calcola in forma chiusa una soluzione $\overline{x}_{i_k}, \overline{x}_{j_k}$ del sotto-problema (2.10) in x_{i_k}, x_{j_k} (ovvero il problema in 2 variabili);

  Sia $x_h^{k+1} = \begin{cases} \overline{x}_h & \text{se } h \in W_k \\ x_h^k & \text{altrimenti} \end{cases}$;

  Sia $\nabla f(x^{k+1}) = \nabla f(x^k) + (x_{i_k}^{k+1} - x_{i_k}^k)Q_{i_k} + (x_{j_k}^{k+1} - x_{j_k}^k)Q_{j_k}$;

  Sia $k = k + 1$;

return x^k

Algoritmo 4: Sequential Minimal Optimization (SMO) per problemi StQP

Da questo pseudo-codice è possibile quindi notare che un punto cardine è la scelta di un corretto punto iniziale x^0 da cui far partire l'algoritmo di decomposizione veloce. Naturalmente questo punto deve essere ammissibile e rispettare il vincolo di simplesso.

L'algoritmo di decomposizione di tipo SMO ha quindi come obiettivo, ad

ogni iterazione, di scendere lungo una direzione ammissibile che abbia solo due componenti diverse da zero in modo da garantire che il valore obiettivo decresca.

2.4.2 Scelta del criterio di arresto

Per scegliere correttamente il criterio d'arresto per l'algoritmo 4 è necessario inizialmente richiamare la condizione di ottimalità del primo ordine per il problema (1.1).

Proposizione 2.5 *Sia $x^\star \in \Delta_n$ un minimo locale del problema (1.1). Allora vale che:*

$$x_j^\star > 0 \quad \Rightarrow \quad \frac{\partial f(x^\star)}{\partial x_j} \leq \frac{f(x^\star)}{\partial x_i} \text{ per ogni } i = 1, \ldots, n \tag{2.11}$$

Un punto ammissibile $x^\star$ che soddisfa dunque la condizione (2.11) è un punto stazionario per il problema (1.1). Sia allora:

- $m(x^\star) = \min_{i=1,\ldots,n} Q_i^T x^\star + c_i$.
- $M(x^\star) = \max_{j:x_j^\star>0} Q_j^T x^\star + c_j$.

la condizione (2.11) può essere equivalentemente riscritta come segue:

$$M(x^\star) \leq m(x^\star) \tag{2.12}$$

Quindi un criterio di arresto adatto per l'algoritmo SMO può essere il seguente:

$$M(x^k) \leq m(x^k) + \epsilon \quad \text{con } \epsilon > 0 \tag{2.13}$$

2.4.3 Selezione del *Working Set*

Le proprietà teoriche e l'efficienza del metodo SMO dipendono dalla regola di selezione *working set*.
Per definire infatti un algoritmo SMO con proprietà di convergenza globale è necessario introdurre il concetto di *most violating pair* per il problema (1.1), ovvero si intende una coppia che viola maggiormente le condizioni di ottimalità.

Definizione 2.6 *Una coppia* $\{i^\star, j^\star\}$ *costituisce una Most Violating Pair (MVP) in* $\overline{x} \in \Delta_n$ *per il problema (1.1) se:*

- $i^\star \in \arg\min_{i=1,\dots,n} Q_i^T \overline{x} + c_i$
- $j^\star \in \arg\max_{j:\overline{x}_j>0} Q_j^T \overline{x} + c_j$

È possibile notare quindi che selezionando nell'algoritmo 4 una *violating pair* (i_k, j_k) come *working set* W e tenendo in considerazione la condizione necessaria di ottimalità di primo ordine per i problemi StQP (2.11), la funzione obiettivo decresce strettamente, ovvero $f(x^{k+1}) < f(x^k)$.
Inoltre è possibile osservare che l'algoritmo 4, con una selezione del working set come *most violating pair*, soddisfa tale condizione (2.13) in un numero finito di iterazioni.

2.4.4 Risolvere il sotto-problema quadratico in due variabili

Dato il *working set* $W = \{i, j\}$ e il suo complementare rispetto a $\{1, \ldots, n\}$, $\overline{W}$, il sotto-problema (2.10) rispetto a x_i, x_j, fissate tutte le altre variabili a $x^k_{\overline{W}}$ è il seguente:

$$\begin{aligned} \min_{x_i, x_j} \quad & \frac{1}{2} \begin{bmatrix} x_i & x_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} + \begin{bmatrix} p_i & p_j \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} \\ \text{s.t.} \quad & x_i + x_j = s \\ & x_i, x_j \geq 0 \end{aligned} \tag{2.14}$$

dove:

- $p = c_W + Q_{W\overline{W}} x^k_{\overline{W}}$
- $s = 1 - \sum_{h \in \overline{W}} x^k_h$

Dato quindi un punto ammissibile $\overline{x}$ e una direzione d di discesa ($\nabla f(\overline{x})^T d < 0$), indichiamo $\overline{\beta}$ il massimo passo ammissibile lungo d a partire da $\overline{x}$.
La procedura per il calcolo della soluzione analitica diventa quindi [21]:

Data: Sia $\overline{\beta} = \overline{x}_j$

1 **if** $(d^\star)^T Q d^\star \leq 0$ **then**

2 $\quad \beta^\star = \overline{\beta}$;

3 **else**

4 $\quad \beta_{nv} = \frac{-\nabla f(\overline{x})^T d^\star}{(d^\star)^T Q d^\star}$;

5 $\quad \beta^\star = min(\overline{\beta}, \beta_{nv})$;

6 Calcola $x^\star = \overline{x} + \beta^\star d^\star$;

7 **return** $x^\star$

Algoritmo 5: Procedura di calcolo della soluzione analitica

La disuguaglianza nel controllo al passo 1 della procedura di calcolo della soluzione analitica, permette di trattare il caso di una matrice non semi-definita positiva allo stesso modo con cui viene trattato il caso di una matrice semi-definita positiva con determinante uguale a 0, garantendo la diminuzione stretta del valore della funzione obiettivo.
La procedura per risolvere tale sotto-problema (2.14), quando W è costituito da una *violating pair*, prende quindi spunto da quella proposta per la risoluzione della classe dei problemi SVM con kernel indefinito [12].

2.4.5 Modifiche per la convergenza globale

Per definire un algoritmo di decomposizione convergente globalmente (in particolare per questo lavoro di tesi SMO), sono necessari tre elementi principali:

1. Il *working set* deve essere scelto da una regola Gauss-Southwell, ovvero una regola che prevede ad ogni iterazione l'aggiornamento di un solo blocco di variabili.

2. La distanza Euclidea tra due iterazioni successive dovrebbe tendere a zero ovvero $\|x^{k+1} - x^k\| \to 0$.

3. I sotto-problemi, che possono essere convessi o non convessi, dovrebbero essere risolti almeno con l'ottimalità locale.

Il punto (1) è soddisfatto dalla regola di selezione di una *Most Violating Pair*.

Il punto (2) è garantito invece dalla "riduzione sufficiente" della funzione obiettivo ad ogni iterazione.

Come viene analizzato nel paper [5], per garantire questa proprietà è necessario modificare il sotto-problema ogni volta che la quantità $Q_{ii} - 2Q_{ij} + Q_{jj}$ è zero.

In particolare in questo caso si deve aggiungere alla funzione obiettivo del sotto-problema (2.14) un termine *proximal-point* così fatto:

$$\frac{\rho}{2}\|x_W - x_W^k\|^2 \text{ con } \rho > 0 \tag{2.15}$$

La funzione obiettivo del sotto-problema (2.14) diventa:

$$(Q_j^T x^k + c_j - Q_i^T x^k - c_i)d_j + \rho\|d_j\|^2 \tag{2.16}$$

Infine per soddisfare il punto (3) è necessario utilizzare una regola di aggiornamento adeguata come segue [5]:

$$d_j^\star = \begin{cases} \max\{-x_j^k, \tilde{d}_j\} & \text{se } Q_{ii} - 2Q_{ij} + Q_{jj} > 0 \\ \max\left\{-x_j^k, -\frac{Q_j^T x^k + c_j - Q_i^T x^k - c_i}{\rho}\right\} & \text{se } Q_{ii} - 2Q_{ij} + Q_{jj} = 0 \\ -x_j^k & \text{se } Q_{ii} - 2Q_{ij} + Q_{jj} < 0 \end{cases} \tag{2.17}$$

Con $\tilde{d}_j = -\frac{Q_j^T x^k + c_j - Q_i^T x^k - c_i}{Q_{ii} - 2Q_{ij} + Q_{jj}}$.

L'algoritmo 4, dotato di una regola di selezione del *working set* come MVP e con questa piccola modifica mostrata in (2.17) può essere teoricamente caratterizzato.

Vale dunque la condizione di "riduzione sufficiente" che è stata accennata precedentemente:

Lemma 2.1 *Sia* $\{x^k\}$ *una sequenza generata dall'algoritmo 4 dotata di una regola di selezione del working set basata su MVP e di una regola di aggiornamento come in (2.17). Allora esiste un* $\sigma > 0$ *tale che per ogni* k *vale*

$$f(x^{k+1}) \leq f(x^k) - \frac{\sigma}{2}\|x^{k+1} - x^k\|^2 \tag{2.18}$$

La dimostrazione di tale lemma è descritta in [5].

Capitolo 3

Grafo di Convessità per problemi StQP (SMO-CG)

In questa sezione si fornisce la descrizione della tecnica che è stata implementata per sviluppare questo lavoro di tesi.
Come accennato nel precedente capitolo un punto fondamentale per utilizzare, nel migliore dei modi, l'algoritmo SMO per i problemi StQP, è scegliere un adeguato punto di partenza. Sono stati eseguiti vari esperimenti, come ad esempio utilizzare un metodo *Iterated Local Search* come perturbazione del vettore di partenza. In particolare una procedura di tipo *Iterated Local Search* esegue varie chiamate di una ricerca locale in cui il vettore di partenza di un iterazione è una qualsiasi perturbazione del vettore della soluzione ottima trovata fino a quel momento. Nessuna di queste prove però ha prodotto un notevole miglioramento rispetto alla letteratura attuale.
Un'idea invece, che ha dato ottimi risultati in termini di prestazioni, è stata quella di sfruttare il grafo di convessità sottostante alla matrice Hessiana di partenza del problema.

3.1 Osservazioni preliminari sui problemi StQP

Data la struttura dei problemi StQP (1.1) è possibile osservare le seguenti affermazioni [22]:

- **Osservazione 3.1** *La matrice Q e il vettore c del problema possono essere sempre ridefiniti in modo che $Q_{ii} = 0$ per ogni $i = 1, \ldots, n$, ovvero tutti gli elementi sulla diagonale della matrice Hessiana Q possono essere posti uguali a zero.*

- **Osservazione 3.2** *Se vale la condizione della diagonale della proposizione 3.1, allora tutti i $Q_{ij} > 0$ possono essere posti uguali a zero, poiché ad ogni soluzione ottima del problema StQP, $Q_{ij} > 0$ implica che almeno una tra x_i e x_j è certamente uguale a zero.*

Si procede adesso alla dimostrazione di queste due osservazioni [23].

Dimostrazione:

Per quanto riguarda la prima parte della dimostrazione (Osservazione 3.1), poiché $\sum_{j=1}^{n} x_j = 1$, la funzione obiettivo può essere riscritta come

$$\sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j - \sum_{i=1}^{n} Q_{ii}x_i \left(\sum_{j=1}^{n} x_j\right) + \sum_{i=1}^{n} Q_{ii}x_i + \sum_{i=1}^{n} c_i x_i,$$

o, equivalentemente

$$\sum_{i=1}^{n}\sum_{j=1,\, j\neq i}^{n} (Q_{ij} - Q_{ii})x_i x_j + \sum_{i=1}^{n}(Q_{ii} + c_i)x_i.$$

Per la seconda parte della dimostrazione (Osservazione 3.2), assumendo che $Q_{ij} > 0$ e, per contraddizione, che esista una soluzione ottima $x^\star$ del problema StQP con $x_i^\star, x_j^\star > 0$. Il valore ottimo è

$$\sum_{i=1}^{n}\sum_{j=1, j\neq i} Q_{ij}x_i^\star x_j^\star + \sum_{i=1}^{n}(Q_{ii} + c_i)x_i.$$

Adesso se vengono fissate tutte le variabili del problema (1.1) eccetto x_i, x_j, si ha

$$\begin{aligned} \min \quad & Q_{ij}x_ix_j + c_i^{'}x_i + c_j^{'}x_j + const \\ \text{s.t.} \quad & x_i + x_j = x_i^\star + x_j^\star \\ & x_i, x_j \geq 0, \end{aligned}$$

e quindi, dopo aver eliminato x_j

$$\begin{aligned} \min \quad & -Q_{ij}x_i^2 + c_i^{''}x_i + const \\ \text{s.t.} \quad & 0 \leq x_i \leq {x_i}^{star} + x_j^\star. \end{aligned}$$

Poiché la funzione obiettivo di questo problema è concava dovuta a $Q_{ij} < 0$, il suo valore minimo viene raggiunto quando $x_i = 0$ o $x_i = x_i^\star + x_j^\star$ (in alternativa $x_j = 0$), contraddicendo così l'assunzione che entrambi x_i e x_j fossero strettamente positivi. □

3.2 Grafo di convessità

Per introdurre il concetto di grafo di convessità, è necessario assumere che gli elementi sulla diagonale della matrice Q siano nulli e che gli elementi al di fuori della diagonale siano nulli o negativi.
Un grafo $G = (V, E)$ può essere associato al problema StQP (1.1), dove:

- $V = \{1, \ldots, V\}$ sono i vertici del grafo.
- $E = \{(i, j) : i, j \in \{1, \ldots, n\}, Q_{ij} < 0\}$ sono gli archi del grafo.

Questo grafo è chiamato *grafo di convessità* del problema (1.1) [24].
Il grafo di convessità è un grafo in cui ad ogni variabile del problema viene associato un vertice e viene posto un arco (i, j) se e solo se la funzione obiettivo è strettamente convessa su tutto lo spigolo che, nell'insieme ammissibile,

congiunge il vertice x_i con il vertice x_j. La funzione è strettamente convessa su un determinato spigolo se vale la seguente condizione: $Q_{ii}+Q_{jj}-2Q_{ij} > 0$. Si definisce adesso il concetto di *clique* e *insieme indipendente*:

Definizione 3.3 *Una clique è un insieme C di vertici in un grafo non orientato G, tale che, per ogni coppia di vertici in C, esiste un arco che li collega.*

Definizione 3.4 *Un insieme indipendente I è un insieme di vertici in un grafo G, nessuno dei quali è adiacente a due a due, ovvero è un insieme I di vertici tale che per ogni due vertici in I non c'è nessun arco che li connette.*

La soluzione ottima del problema StQP (1.1), come viene analizzato nel paper [24], deve essere raggiunta nell'interno relativo di una faccia:

$$F_C = \{x \in \Delta_n \, : \, x_i = 0, i \notin C\} \tag{3.1}$$

dove $C \subseteq V$ è una *clique* sul grafo di convessità G.
Quanto detto implica dunque la seguente proprietà:

Proprietà 3.5 *Dato un qualsiasi insieme indipendente $I \subseteq V$ del grafo di convessità G, al massimo un x_i, $i \in I$, è strettamente positivo alla soluzione ottima del problema (1.1).*

Questo significa che in una soluzione ottimale se esistono più componenti diverse da zero, queste devono stare in una *clique*.
Le condizioni KKT per il problema (1.1) sono le seguenti [6]:

$$\begin{aligned} q_i x + c_i - \lambda &= \mu \quad i = 1, \ldots, n \\ \mu_i &\geq 0 \quad i = 1, \ldots, n \end{aligned} \tag{3.2}$$

dove:

- q_i è l'i-esima riga della matrice Q.
- λ è il moltiplicatore di Lagrange del vincolo $\sum_{i=1}^{n} x_i = 1$.
- μ_i con $i = 1, \ldots, n$ è il moltiplicatore di Lagrange del vincolo $x_i \geq 0$.

La ricerca quindi della soluzione ottima per il problema (1.1) può essere ristretta ai punti KKT.
Questa proprietà 3.5 può essere sfruttata, nell'ottica di un algoritmo di ottimizzazione globale, utilizzando un *solver* locale, per dare una soluzione iniziale che la soddisfi.

3.3 Strategia di inizializzazione dei punti per un algoritmo multi-start

La struttura di un algoritmo SMO multi-start per la classe dei problemi StQP è riportata nel seguente pseudo-codice:

```
1 Sia x^0 ∈ Δ_n un punto di partenza;
2 sol = SMO per StQP (Algoritmo 4);
3 Sia m il numero di iterazioni da eseguire;
4 for i = 1 ... m do
5 |   Sia x^0 ∈ Δ_n un altro punto di partenza;
6 |   tmp = SMO per StQP (Algoritmo 4);
7 |   if tmp < sol then
8 |   |   sol = tmp;
9 return sol
```

Algoritmo 6: SMO Multi-Start per StQP

L'algoritmo multi-start confronta $(m + 1)$ risultati ottenuti utilizzando l'algoritmo 4 SMO per StQP $(m+1)$ volte, ogni volta partendo da un punto iniziale ammissibile x^0 diverso.
In un algoritmo multi-start quindi è fondamentale una corretta scelta dei punti iniziali da cui far partire la procedura di ottimizzazione.
Una prima soluzione è stata quella di utilizzare come punti di partenza:

$$x^i = (0, 0, \ldots, 1, \ldots, 0, 0) \text{ con } i = 1, \ldots, n \tag{3.3}$$

dove il valore 1 viene assegnato all'indice $i - esimo$ mentre tutte le altre componenti sono poste a 0. Dai risultati del paper [5] è possibile osservare però che tale strategia non ha portato a soluzioni ottimali.
Un'idea quindi è stata quella di sfruttare l'informazione relativa al grafo di convessità. Dal momento che però calcolarsi le *clique* è un problema difficile tanto quanto risolvere il problema di partenza, una buona strategia è cercare soluzioni iniziali in cui solo due componenti sono diverse da zero.
In una soluzione ottimale infatti queste due componenti possono essere contemporaneamente diverse da zero solo se sono connesse nel grafo di convessità. Si sceglie quindi la prima componente in maniera casuale e poi si sceglie sempre casualmente un'altra delle variabili che sono connesse, con tale componente, nel grafo di convessità.

3.4 Costruzione della matrice binaria

Per selezionare quindi un efficiente punto di partenza per l'algoritmo SMO (**SMO-CG**) come analizzato nella sezione 3.3 è necessario costruire la matrice binaria relativa al grafo di convessità della matrice Hessiana Q di partenza. Tale matrice binaria è utile perchè indica se due variabili sono compatibili o

meno nella soluzione ottimale.

La sua costruzione avviene tramite il seguente algoritmo:

Data: Sia la matrice Hessiana $Q \in \mathbb{R}^{n \times n}$

Sia una matrice $H \in \mathbb{R}^{n \times n}$ dove $H_{ij} = 0 \ \forall i, j$;
for $i = 1 \dots n$ **do**
  for $j = 1 \dots n$ **do**
    if $i \neq j$ **then**
      $H_{ij} = \frac{2Q_{ij} - Q_{ii} - Q_{jj}}{2}$;
      if $H_{ij} > 0$ **then**
        $H_{ij} = 0$;
      if $H_{ij} < 0$ **then**
        $H_{ij} = 1$;
$H_{ii} = 0$ con $i = 1, \dots, n$;
return H

Algoritmo 7: Costruzione della matrice binaria

Come è possibile osservare dall'algoritmo 7 viene quindi inizializzata una matrice quadrata vuota H di dimensione $n \times n$ come la matrice Hessiana di partenza Q.

Ogni elemento extra-diagonale della matrice H viene calcolato con la formula $H_{ij} = \frac{2Q_{ij} - Q_{ii} - Q_{jj}}{2}$. Una volta eseguito questo passaggio se l'elemento extra-diagonale risultante assume un valore maggiore di zero gli viene assegnato il valore 0, altrimenti se assume un valore minore di zero gli viene assegnato il valore 1. Infine come ultimo passaggio tutti i valori della matrice sulla diagonale vengono posti a 1.

Tale procedura quindi rende in uscita la matrice binaria H relativa alla matrice Hessiana Q di partenza.

3.4.1 Scelta del punto di partenza

Una volta ottenuta quindi la matrice binaria costruita come analizzato nel paragrafo precedente 3.4, è necessario stabilire come scegliere in maniera efficiente il punto di partenza da cui far partire l'algoritmo di decomposizione SMO adattato ai problemi StQP.
Dopo aver eseguito diverse combinazioni, la migliore procedura che ha dato ottimi risultati, come riportato nel capitolo successivo 4, è quella schematizzata nel seguente pseudo-codice:

Data: Sia la matrice binaria $H \in \mathbb{R}^{n \times n}$

Sia un vettore $x \in \mathbb{R}^n$ con $x_i = 0$ con $i = 1, \ldots, n$;

Estrai a caso $i \sim \mathcal{U}(1, \ldots, n)$;

$J = \{j | H_{ij} = 1\}$;

$s = length(J)$;

Estrai a caso $k \sim ([0, \ldots, s-1])$;

$\hat{j} = J[k]$;

$x = \left[0, 0, 0, \ldots, \frac{1}{2}, 0, 0, \ldots, \frac{1}{2}, 0, \ldots, 0\right]$ dove $x[i] = \frac{1}{2}$ e $x[\hat{j}] = \frac{1}{2}$;

return x

Algoritmo 8: Scelta del punto di partenza

Il vettore di partenza, ottenuto tramite questa procedura, è composto da tutti zeri tranne che per gli indici i e $\hat{j}$ a cui invece è assegnato il valore $\frac{1}{2}$. In questo modo quindi il vincolo di simplesso della formulazione dei problemi StQP viene rispettato ed è una scelta efficiente da utilizzare all'interno dell'algoritmo SMO.
La scelta di eseguire due selezioni casuali all'interno dell'algoritmo 8 per scegliere prima l'indice i e poi quello relativo a $\hat{j}$ è stata fatta perchè produceva soluzioni migliori, infatti seguire un ordine, oltre ad essere computazional-

mente più costoso, non produceva risultati ottimali.
I risultati ottenuti da questa doppia selezione random si potranno vedere nel capitolo 4.
Un aspetto fondamentale quindi da tenere in considerazione è che non è tanto l'aspetto continuo la parte difficile del problema, ma è l'aspetto combinatorio di trovare il supporto, ovvero conoscere quali variabili definiscono la soluzione ottimale e non il loro valore.

Capitolo 4

Esperimenti ed Analisi dei Risultati

In questo capitolo si mostrano i vari esperimenti eseguiti ed i risultati ottenuti affinché si possa valutare all'atto pratico la bontà dell'algoritmo di decomposizione veloce SMO dotato di una strategia multi-start basata sul grafo di convessità sottostante (**SMO-CG**).
Tutti gli esperimenti per questo lavoro di tesi sono stati eseguiti sullo stesso computer, in particolare su una macchina con le seguenti specifiche: CPU Intel Xeon con 6 core e 32 GB di memoria RAM.
Per implementare l'algoritmo SMO ed i vari test è stata utilizzata *NumPy*, una libreria open source per il linguaggio di programmazione Python, che aggiunge supporto a matrici di grandi dimensioni e array multidimensionali, insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati [25].

L'intento dei vari esperimenti svolti è stato quello di:

- Mostrare che l'algoritmo di decomposizione SMO è un valido *local solver* per i problemi StQP.
- Mostrare che l'algoritmo SMO dotato di una strategia multi-start basata sul grafo di convessità sottostante (**SMO-CG**), ha delle ottime performance in termini di efficienza rispetto ad altri approcci allo stato dell'arte.

4.1 Descrizione dei dataset

Per questo studio di tesi sono stati impiegati tre diversi dataset di problemi StQP:

- **Dataset Generic**: composto da 56 istanze di problemi StQP, generati da Nowak nell'articolo [26] e impiegati da Liuzzi in [27]. La dimensione n dei problemi varia tra $\{10, 30, 50, 100, 200, 500, 1000\}$ e anche la densità del grafo di convessità sottostante varia tra $\{0.25, 0.5, 0.75, 0.9\}$.
- **Dataset BSU**: un insieme di 20 problemi con dimensione $n \in \{5, \ldots, 24\}$. Sono stati progettati da Bomze [28] per essere "*difficili*", avendo un numero esponenziale di minimi locali stretti.
- **Dataset BHOSLIB**: un insieme di problemi *maximum clique* che si sono dimostrati particolarmente "*difficili*" sia dal punto di vista teorico che pratico [29]. Il dataset è composto da 5 problemi per ciascuna delle 8 dimensioni diverse considerate, ovvero con
  $n \in \{450, 595, 760, 945, 1150, 1272, 1400, 1534\}$.
  Per ogni problema, la dimensione reale della *maximum clique* è nota.

Per quanto riguarda l'impostazione dei parametri, l'unico da scegliere è la tolleranza del criterio di arresto (2.13), che per l'algoritmo SMO è stato impostato pari a $\epsilon = 10^{-8}$ per tutti gli esperimenti su tutti e tre i dataset.

4.2 Analisi dei risultati per il Dataset Generic

Per quanto riguarda il **Dataset Generic** sono stati confrontati tre algoritmi:

1. Branch-and-bound (algoritmo esatto che garantisce l'ottimalità proposto da Liuzzi in [6]).
2. SMO multi-start dove SMO è utilizzato come *local solver*.
3. SMO multi-start basato sul grafo di convessità (**SMO-CG**).

Nel caso di SMO multi-start classico i punti di partenza dell'algoritmo sono stati scelti in modo deterministico e sono $x^i = (0, 0, \ldots, 1, \ldots, 0, 0)$ con $i = 1, \ldots, n$ dove il valore 1 viene assegnato all'indice $i - esimo$ mentre tutte le altre componenti sono poste a 0.
Nel caso invece di SMO-CG sono state eseguite 2000 iterazioni e per ciascuna iterazione il punto di partenza era scelto come descritto nel capitolo 3.
Si confrontano dunque le performance dei tre algoritmi in termini di tempo impiegato per trovare l'ottimo e il valore della funzione obiettivo.
I risultati ottenuti sono riportati nelle seguenti tabelle:

Problema	**Branch&Bound**		**SMO Multi-Start**		**SMO-CG**	
	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo
10x10(0.75)	0.50327	**-4.970562**	0.06247	**-4.970562**	**0.00365**	**-4.970562**
30x30(0.75)	0.71019	**-6.337451**	0.12537	**-6.337451**	**0.01742**	**-6.337451**
50x50(0.25)_1	2.06694	**-5.499346**	0.18439	**-5.499346**	**0.14175**	**-5.499346**
50x50(0.25)_2	0.85614	**-5.275832**	0.19832	**-5.275832**	**0.17750**	**-5.275832**
50x50(0.25)_3	0.58636	**-5.249112**	0.16721	**-5.249112**	**0.00770**	**-5.249112**
50x50(0.25)_4	0.74233	**-4.952773**	0.18030	**-4.952773**	**0.07776**	**-4.952773**
50x50(0.25)_5	0.73594	**-4.775142**	0.15505	**-4.775142**	**0.07386**	**-4.775142**
50x50(0.75)	2.30690	**-6.386014**	0.34134	**-6.386014**	**0.04116**	**-6.386014**
100x100(0-1)	3.30984	0.010527	0.47507	0.010527	**0.64093**	**0.002631**
100x100(0.25)_3	2.85648	**-5.615125**	0.65401	**-5.615125**	**0.57137**	**-5.615125**
100x100(0.5)	13.7352	**-6.279911**	0.73749	**-6.279911**	**0.18496**	**-6.279911**
100x100(0.5)_1	9.90482	**-6.140713**	0.62673	**-6.140713**	0.90827	**-6.140713**
100x100(0.5)_2	1.83253	**-6.277328**	0.65094	**-6.277328**	**0.04339**	**-6.277328**
100x100(0.5)_3	13.0005	**-6.279335**	0.70960	**-6.279335**	**0.19757**	**-6.279335**
100x100(0.5)_4	11.1000	**-6.267527**	0.63115	**-6.267527**	**0.07529**	**-6.267527**
100x100(0.5)_5	20.6519	-6.140713	0.67379	-6.140713	**0.71961**	**-6.180967**
100x100(0.75)	70.0133	**-6.630418**	0.90419	**-6.630418**	**0.30508**	**-6.630418**
100x100(0.75)_1	55.5806	**-6.574007**	1.10742	-6.516996	**0.08798**	**-6.574007**
100x100(0.75)_2	101.576	**-6.539024**	0.94502	**-6.539024**	**0.42413**	**-6.539024**
100x100(0.75)_3	1.82734	**-6.631116**	1.01741	**-6.631116**	**0.65292**	**-6.631116**
100x100(0.75)_4	47.3251	**-6.564830**	0.92951	**-6.564830**	**0.36482**	**-6.564830**
100x100(0.75)_5	42.4800	**-6.813091**	0.94838	**-6.813091**	**0.85671**	**-6.813091**

Tabella 4.1: Risultati per SMO Multi-Start, SMO-CG Multi-Start e Branch&Bound sul Dataset Generic.

Problema	**Branch&Bound**		**SMO Multi-Start**		**SMO-CG**	
	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo
100x100(0.9)_0	2.08992	**-7.014810**	1.04650	**-7.014810**	**0.03042**	**-7.014810**
100x100(0.9)_1	1.96435	**-6.932718**	1.43666	**-6.932718**	**0.09887**	**-6.932718**
100x100(0.9)_2	105.050	**-6.866139**	1.75543	**-6.866139**	**0.08208**	**-6.866139**
100x100(0.9)_3	1.92823	**-6.731264**	0.97941	**-6.731264**	**0.20870**	**-6.731264**
100x100(0.9)_4	65.7972	**-7.148924**	1.01242	**-7.148924**	**0.19978**	**-7.148924**
100x100(0.9)_5	2.14066	**-6.720093**	1.16682	-6.720087	**0.11671**	**-6.720093**
200x200(0.25)	3.63936	**-6.578052**	1.68968	-6.578029	**1.36596**	**-6.578052**
200x200(0.25)_1	28.6062	**-6.099806**	2.04614	**-6.099806**	**0.13210**	**-6.099806**
200x200(0.25)_2	23.7415	**-6.299161**	1.81999	**-6.299161**	**1.32100**	**-6.299161**
200x200(0.25)_3	36.0284	**-6.067600**	1.72144	**-6.067600**	**1.52448**	**-6.067600**
200x200(0.25)_4	17.7363	**-6.455333**	1.65061	**-6.455333**	**0.77820**	**-6.455333**
200x200(0.25)_5	3.30702	**-6.205503**	1.70957	-6.205481	**0.15955**	**-6.205503**
200x200(0.5)	174.710	**-6.879762**	**2.06063**	**-6.879762**	2.28106	**-6.879762**
200x200(0.5)_1	264.872	**-6.434906**	2.16741	**-6.434906**	**0.57370**	**-6.434906**
200x200(0.5)_2	3.89241	**-6.832687**	2.04779	**-6.832667**	**0.50300**	**-6.832667**
200x200(0.5)_3	277.845	**-6.622237**	2.15344	**-6.622237**	**0.90551**	**-6.622237**
200x200(0.5)_4	3.92369	**-6.718093**	2.65052	-6.718074	**0.00673**	**-6.718093**
200x200(0.5)_5	131.274	**-6.727132**	1.94788	**-6.727132**	**1.13313**	**-6.727132**
200x200(0.75)_1	Errore	Errore	3.58427	**-6.704596**	**0.30677**	**-6.704596**
200x200(0.75)_2	Errore	Errore	**3.00780**	**-6.896143**	3.50311	**-6.896143**
200x200(0.75)_3	Errore	Errore	3.35863	**-6.998139**	**0.65663**	**-6.998139**
200x200(0.75)_4	Errore	Errore	3.20046	**-6.838571**	**1.84797**	**-6.838571**
200x200(0.75)_5	Errore	Errore	2.98410	**-6.868261**	**0.36641**	**-6.868261**

Tabella 4.2: continua.

Problema	**Branch&Bound**		**SMO Multi-Start**		**SMO-CG**	
	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo
500x500(0.25)_1	Errore	Errore	9.72614	**-6.813400**	**4.24050**	**-6.813400**
500x500(0.25)_2	Errore	Errore	9.53049	**-6.674184**	**0.62997**	**-6.674184**
500x500(0.25)_3	Errore	Errore	9.84075	**-6.593160**	**4.33393**	**-6.593160**
500x500(0.25)_4	Errore	Errore	9.58323	**-6.400453**	**2.42304**	**-6.400453**
500x500(0.25)_5	Errore	Errore	**9.58552**	-6.580944	13.1758	**-6.667456**
500x500(0.5)_1	Errore	Errore	13.4705	**-6.965748**	**0.33857**	**-6.965748**
500x500(0.5)_2	Errore	Errore	12.6349	-7.073821	**6.81897**	**-7.084148**
500x500(0.5)_3	Errore	Errore	12.6753	**-6.866721**	**1.04319**	**-6.866721**
500x500(0.5)_4	Errore	Errore	**13.1264**	**-6.839946**	15.8508	**-6.839946**
500x500(0.5)_5	Errore	Errore	12.5242	**-7.128226**	**0.20831**	**-7.128226**
1000x1000(0.25)	Errore	Errore	39.5812	**-6.764740**	**20.8814**	**-6.764740**

Tabella 4.3: continua.

In queste tabelle sono evidenziati in grassetto per ogni singolo problema i migliori risultati ottenuti sia per quanto riguarda il tempo per trovare l'ottimo che il valore da esso assunto.

Dai risultati ottenuti è quindi possibile osservare come l'algoritmo SMO-CG multi-start abbia delle prestazioni altamente superiori in termini di tempo per trovare l'ottimo rispetto sia a SMO multi-start che al Branch&Bound. Si parla in alcuni casi anche di molti secondi di differenza specialmente all'aumentare della dimensione dei problemi.

È possibile anche notare che per 16 problemi su 56 (quelli con dimensioni 500 e 1000 e quelli di dimensioni 200 e densità 0,75) l'algoritmo branch-and-bound non è in grado di produrre una soluzione a causa dell'errore *out-of-memory*. SMO-CG in particolare trova l'ottimo anche per quei problemi in cui invece

l'algoritmo SMO multi-start classico non riesce e se confrontato anche con il branch-and-bound si può vedere come sia nettamente più veloce.

4.3 Analisi dei risultati per il Dataset BSU

Per quanto riguarda il **Dataset BSU** sono stati confrontati gli stessi tre algoritmi utilizzati per il Dataset Generic.
Anche la scelta dei punti di partenza di SMO multi-start e SMO-CG è la medesima della sezione precedente (sez. 4.2).
Si confrontano dunque le performance dei tre algoritmi in termini di tempo impiegato per trovare l'ottimo e il valore della funzione obiettivo.
I risultati ottenuti sono riportati nella seguente tabella:

Problema	**Branch&Bound**		**SMO Multi-Start**		**SMO-CG**	
	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo	Tempo per l'Ottimo	Funzione Obiettivo
BSU5	0.51644	**-1.636363**	0.00963	**-1.636363**	**0.00545**	**-1.636363**
BSU6	0.12663	**-0.5**	0.00093	**-0.5**	**5.38e-05**	**-0.5**
BSU7	0.62211	**-4.166666**	0.02591	**-4.166666**	**0.00301**	**-4.166666**
BSU8	0.73491	**-7.25**	Errore	Errore	**0.01283**	**-7.25**
BSU9	0.70521	**-6.666666**	Errore	Errore	**0.01425**	**-6.666666**
BSU10	3.81836	**-13.445576**	0.058673	**-13.445576**	**0.15623**	**-13.445576**
BSU11	0.90817	**-4.077922**	0.18030	**-4.077922**	**0.03516**	**-4.077922**
BSU12	1.05973	**-155.054707**	0.14349	**-155.054707**	**0.00999**	**-155.054707**
BSU13	1.25181	**-10.8**	0.30676	**-10.8**	**0.13393**	**-10.8**
BSU14	0.82476	**-2.575757**	0.54084	**-2.575757**	**0.02220**	**-2.575757**
BSU15	3.14174	**-6.634632**	22.53217	**-6.634632**	**1.32363**	**-6.634632**
BSU16	0.95529	**-4.518292**	0.40670	**-4.518292**	**0.02435**	**-4.518292**
BSU17	5.21040	**-951.686783**	3.43359	**-951.686783**	**2.36103**	**-951.686783**
BSU18	1.22495	**-5.57451**	1.63602	**-5.57451**	**0.08308**	**-5.57451**
BSU19	2.4975	**-22.436282**	0.80794	**-22.436282**	**0.05586**	**-22.436282**
BSU20	1.46311	**-10.20502**	1.76443	**-10.20502**	**0.07474**	**-10.20502**
BSU21	1.66949	**-11.320754**	0.55238	**-11.320754**	**0.02818**	**-11.320754**
BSU22	2.57263	**-3.091891**	6.48831	**-3.091891**	**0.28407**	**-3.091891**
BSU23	116.476	**-19881.060**	191.978	**-19881.060**	**52.4680**	**-19881.060**
BSU24	679.746	**-11.445673**	6.19012	**-11.445673**	**0.24596**	**-11.445673**

Tabella 4.4: Risultati per SMO Multi-Start, SMO-CG Multi-Start e Branch&Bound sul Dataset BSU.

Dai risultati ottenuti è possibile osservare che tutti e tre gli algoritmi trovano sempre l'ottimo, però SMO-CG ha prestazioni in termini di tempo molto superiori. Ad esempio per quanto riguarda il problema *BSU23*, SMO-CG è più veloce a trovare la soluzione ottima di quasi 50 secondi rispetto

agli altri due algoritmi.
SMO-CG inoltre riesce a trovare la soluzione anche quando invece SMO multi-start fallisce come ad esempio per i problemi *BSU8* e *BSU9*.
In generale comunque per quasi tutti i problemi di questo dataset l'algoritmo SMO-CG trova la soluzione ottima in un tempo nell'ordine del decimo di secondo, il che denota appunto la bontà di tale algoritmo.

4.4 Analisi dei risultati per il Dataset BHOSLIB

Per quanto riguarda il **Dataset BHOSLIB** sono stati confrontati solo due su tre algoritmi: SMO multi-start e SMO-CG. Il metodo branch-and-bound infatti non è stato possibile utilizzarlo perchè non riusciva a produrre una soluzione a causa della grandezza e complessità dei problemi.
Nel caso di SMO multi-start classico i punti di partenza dell'algoritmo sono stati scelti in modo deterministico e sono $x^i = (0, 0, \ldots, 1, \ldots, 0, 0)$ con $i = 1, \ldots, n$ dove il valore 1 viene assegnato all'indice $i - esimo$ mentre tutte le altri componenti sono poste a 0.
Nel caso invece di SMO-CG sono state eseguite 4000 iterazioni e per ciascuna iterazione il punto di partenza era scelto come descritto nel capitolo 3.
In questo esperimento per conoscere la dimensione della *clique* ottenuta dopo aver applicato uno dei due algoritmi è stato necessario trasformare il valore della funzione obiettivo tramite la seguente formula:

$$clique\_size = \frac{1}{1 + f^\star} \quad \text{dove } f^\star \text{ è la soluzione ottima} \tag{4.1}$$

Per ogni valore del numero di vertici e dimensione della *maximum clique*, vengono riportate la migliore soluzione trovata per ciascuna delle 5 istanze

dei problemi, il tempo medio per trovare l'ottimo e la sua deviazione standard (quest'ultima calcolata solo nel caso dell'algoritmo SMO-CG).
I risultati ottenuti sono riportati nelle seguenti tabelle:

Vertici	**Dim. Max Clique**	**P1 sol**	**P2 sol**	**P3 sol**	**P4 sol**	**P5 sol**	**Avg. Tempo per l'Ottimo**
450	30	**26**	26	25	25	**26**	**35.7024**
595	35	30	**30**	29	31	30	**76.8016**
760	40	33	33	**34**	**34**	33	**130.8810**
945	45	37	35	35	37	37	**157.0915**
1150	50	40	39	**43**	40	41	337.2827
1272	53	**43**	45	43	44	**45**	526.0112
1400	56	46	45	45	45	44	744.5845
1534	59	48	47	46	48	48	1042.0672

Tabella 4.5: Risultati per SMO Multi-Start sui problemi maximum clique difficili del Dataset BHOSLIB.

Vertici	**Dim. Max Clique**	**P1 sol**	**P2 sol**	**P3 sol**	**P4 sol**	**P5 sol**	**Avg. Tempo per l'Ottimo**	**Std. Tempo per l'Ottimo**
450	30	25	26	**26**	**26**	25	19.014	17.482
595	35	30	29	**30**	31	**31**	133.822	121.183
760	40	33	**35**	33	33	**34**	164.536	135.605
945	45	37	**37**	**36**	37	**38**	170.262	192.360
1150	50	40	**41**	42	40	41	**322.639**	272.615
1272	53	42	45	**44**	44	42	**307.074**	260.703
1400	56	46	45	45	**46**	**46**	**404.924**	299.221
1534	59	48	47	**48**	48	48	**936.406**	789.548

Tabella 4.6: Risultati per SMO-CG sui problemi maximum clique difficili del Dataset BHOSLIB.

In grassetto sono evidenziati i risultati in cui un algoritmo è migliore dell'altro. È possibile quindi osservare che SMO-CG all'aumentare della dimensione dei problemi risulta avere migliori prestazioni, specialmente per quanto riguarda il tempo per arrivare alla soluzione ottima.
SMO-CG inoltre ottiene 14 vittorie sull'algoritmo SMO multi-start, che a sua volta ottiene 8 vittorie nei confronti di SMO-CG. In particolare nei due problemi di dimensione maggiore ($n = 1400$ e $n = 1534$) SMO-CG ottiene 3 vittorie e sui restanti problemi pareggia i risultati ottenuti con SMO multi-start.
Dai dati ottenuti, possiamo quindi trarre alcune conclusioni interessanti. È evidente che, con problemi particolarmente difficili, un banale multi-start deterministico non è sufficiente per arrivare all'ottimale globale. Tuttavia, l'algoritmo SMO-CG multi-start è in grado di recuperare soluzioni abbastanza buone in un ragionevole quantità di tempo.

Capitolo 5

Conclusioni e Sviluppi Futuri

In questo lavoro abbiamo mostrato come l'algoritmo Sequential Minimal Optimization (SMO), utilizzato frequentemente per risolvere problemi di training di SVM non lineari, può essere facilmente adattato per gestire l'importante classe dei problemi StQP non convessi.
In particolare è stata implementata una tecnica che permettesse di utilizzare in modo efficiente SMO come *local solver* all'interno di una procedura di ottimizzazione globale.
Un aspetto fondamentale è stato quello di scegliere un punto di partenza adeguato per l'algoritmo. Sono state provate varie tecniche per la scelta di tale punto, ma quella che ha dato migliori risultati è stata quella che utilizza il grafo di convessità associato al problema di partenza.
Tramite gli esperimenti effettuati è possibile affermare che SMO-CG, implementato in questo lavoro di tesi, può effettivamente essere utilizzato come un ingranaggio chiave inserito in un contesto di ottimizzazione globale per risolvere più velocemente i problemi StQP all'ottimo globale.
SMO-CG inoltre si è rivelato migliore, in termini di prestazioni, rispetto a SMO equipaggiato con una semplice strategia multi-start, specialmente al-

l'aumentare della dimensione dei problemi StQP.

Lo sviluppo dell'algoritmo SMO-CG multi-start potrebbe quindi portare a preferire il suo utilizzo al posto di approcci complessi come il branch-and-bound, come ad esempio quello proposto da Liuzzi in [6], che assicurano di trovare il minimo globale, però a fronte di un tempo computazionale molto elevato.

Un possibile sviluppo futuro potrebbe essere quello di utilizzare una regola di selezione delle variabili diversa, come ad esempio la regola che utilizza le informazioni del secondo ordine, affinché il processo possa essere velocizzato e si possa trovare anche soluzioni migliori [30].

Un altro sviluppo futuro potrebbe essere quello di impiegare strategie che utilizzino in maniera più raffinata il grafo di convessità, tenendo però sempre in considerazione il tempo impiegato per selezionare il punto di partenza, affinché questo non diventi superiore alla risoluzione del problema stesso.

Bibliografia

[1] I. M. Bomze, "On standard quadratic optimization problems," 1998.

[2] D. Ho, "Quadratic programming for portfolio optimization," 1992. `https://www.mathworks.com/help/optim/ug/quadratic-programming-portfolio-optimization-problem-based.html`.

[3] R. Horst, P. Pardalos, and N. VanThoai, "Introduction to global optimization," 2000.

[4] Hosch and L.William, "P versus np problem," 2020.

[5] R. Bisori, M. Lapucci, and M. Sciandrone, "A study on sequential minimal optimization methods for standard quadratic problems," 2021.

[6] G. Liuzzi, M. Locatelli, and V. Piccialli, "A new branch-and-bound algorithm for standard quadratic programming problems," *Optimization Methods and Software*, 2019.

[7] Markowitz, "Portfolio selection," 1952.

[8] T. Motzkin and E. Straus, "Maxima for graphs and a new proof of a theorem of turán," 1965.

[9] I. M. Bomze, "Standard quadratic optimization problems: Applications," 2008.

[10] J. Kingman, "A mathematical problem in population genetics," 1961.

[11] I. M. Bomze, "Regularity versus degeneracy in dynamics, games, and optimization: A unified approach to different aspects," 2002.

[12] H. Lin and J. Lin, "A study on sigmoid kernels for svm and the training of nonpsd kernels by smo-type methods," 2003.

[13] E. de Klerk and V. Pasechnik, "A linear programming reformulation of the standard quadratic optimization problem," 2006.

[14] S. Burer and D. Vandenbussche, "A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxions," 2008.

[15] J. Chen and S. Burer, "Globally solving nonconvex quadratic programming problems via completely positive programming," 2012.

[16] I. M. Bomze, "Branch-and-bound approaches to standard quadratic optimization problems," 2002.

[17] J. Hu, E. Mitchell, and J. Pang, "An lpcc approach to nonconvex quadratic programs," 2012.

[18] V. Vapnik, "Pattern recognition using generalized portrait method," 1963.

[19] Platt and John, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[20] C. Chang, C. Hsu, and C. Lin, "The analysis of decomposition methods for support vector machines," 2000.

[21] R. Bisori, "Metodi di tipo smo per standard quadratic programming," 2018.

[22] M. Locatelli, "Convex envelopes of some quadratic functions over the n-dimensional unit simplex," 2015.

[23] M. Locatelli, "Computing convex envelopes of quadratic functions over the unit simplex," 2014.

[24] A. Scozzari, F. Tardella, and H. Takahashi, "A clique algorithm for standard quadratic programming," 2008.

[25] T. Oliphant, "Guide to numpy," 2006.

[26] I. Nowak, "A new semidefinite programming bound for indefinite quadratic forms over a simplex," 1999.

[27] G. Liuzzi, M. Locatelli, and V. Piccialli, "Branch & bound for stqp," `http://www.iasi.cnr.it/~liuzzi/StQP/`.

[28] I. M. Bomze, "The complexity of simple models-a study of worst and typical hard cases for the standard quadratic optimization problem," 2018.

[29] K. Xu, F. Boussemart, F. Hemery, and C. Lecoutre, "Random constraint satisfaction: easy generation of hard (satisfiable) instances," 2007. `http://sites.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.html`.

[30] R. Fan, P. Chen, and J. Lin, "Working set selection using second order information for training support vector machines," 2005.

Ringraziamenti

Non avrei mai pensato di trovarmi a scrivere dei ringraziamenti, ma arrivati a conclusione del percorso universitario ritengo sia giusto dedicare due righe alle persone che mi hanno affiancato in tutti questi anni. Innanzitutto ringrazio il professor Fabio Schoen e il dottore Matteo Lapucci per avermi seguito ed essere sempre stati a disposizione durante questo lavoro di tesi. Un grandissimo ringraziamento va alla mia famiglia, per avermi sempre supportato e per avermi dato l'opportunità di seguire questo percorso di studi. In particolare voglio ringraziare mia sorella (la mia *Sister*) per essermi sempre stata vicino, grazie. Ed ora passiamo agli amici, un grazie a tutto il gruppo Antella & Co. (chiamato così anche se per metà non vive all'Antella) di cui faccio i nomi in rigoroso ordine alfabetico: Berna, Edo, Gianne, Gio Bertini & Family, Gio Malik, Gréta, Matte, Marghe, Piro, Piter, Ricca, Samu, Sara, Simo, Trapu. Un altro doveroso ringraziamento va invece al gruppo Università (Caddo, Cecche, Lollo, Nedo) con cui abbiamo condiviso un percorso, a tratti tortuoso, ma che non ci ha impedito di raggiungere ognuno i propri traguardi. Un altro grande ringraziamento va agli amici del Fanta (Ale, Andre, Cingo, Fritta, Karam, Sansa, Sori), con cui ogni anno ci battagliamo a suon di rilanci e gufate preventive, vi voglio bene. Ultimi ma non meno importanti i ringraziamenti ai miei compagni di squadra del Firenze2Basket di cui sono orgogliosamente *"Mirco 18 Il Capitano"*, con cui ho condiviso e condivido tuttora la mia più grande passione: il Basket. A tutti voi voglio dirvi grazie di cuore e dedicarvi questa piccola poesia:

"Si in terrarum orbem fuisse paulum bonitatem
et unusquisque fratrem suum considerasse,
ibi esset minus cogitationes minusque dolores
et mundus esset consequuntur quanto magis pulcher."

P.P.