Università degli Studi di Padova
Dipartimento di Matematica Tullio Levi-Civita

Tesi di Laurea Triennale in Matematica

Algoritmi di Las Vegas per Programmi Lineari di dimensione piccola

Relatore:
Prof. Di Summa Marco

Candidato:
Fozzati Antonio

Anno Accademico 2021/2022

Indice

Capitolo 1

Premessa

In questa tesi saranno analizzati tre articoli [3], [7], [11] che hanno come scopo la costruzione di un algoritmo capace di risolvere Programmi Lineari in tempi lineari rispetto al numero di vincoli. Il principale algoritmo utilizzato per risolvere dei Programmi Lineari (PL) è il metodo del simplesso; è noto però come tutte le regole anticiclaggio conosciute finora (compresa quella di Bland) abbiano un tempo di calcolo non polinomiale ma esponenziale; per ognuna si possono infatti trovare casi per cui sono necessarie circa 2^d iterazioni del metodo (dove d è il numero di variabili del problema); questa inefficienza teorica ha portato alla costruzione di altri metodi per la soluzione dei PL come il metodo dell'ellissoide o i metodi di punto interno. Entrambi questi algoritmi sono polinomiali, l'ellissoide è decisamente inefficiente sul lato pratico rispetto al metodo del simplesso, mentre i metodi di punto interno sono molto utilizzati.

La domanda che N. Megiddo si pone è se sia possibile creare degli algoritmi polinomiali nel numero di vincoli del PL (n) nel caso in cui il numero di variabili sia molto piccolo (costante o limitato da una funzione che cresce molto lentamente in n). Risolve come prima cosa questo problema nei casi d=1,2 [8], non trattati in questo testo, per poi espandere tale risultato a d generici, ma sempre piccoli, ottenendo un bound pari a $O(2^{2^d} n)$, lineare in n come auspicato, ma superesponenziale in d, che lo rende scomodo anche per d piccoli.

L'intuizione di Dyer e Frieze [6] di introdurre una componente aleatoria all'interno dell'algoritmo migliora questo bound, ma il miglior risultato ottenuto in questo testo sarà quello di K. Clarkson che porterà un bound pari a $O(d^2 n) + (\log n) O(d)^{\frac{d}{2}+O(1)} + O(d^4 \sqrt{n} \log n)$, in cui la componente dominante rispetto ad n ha una dipendenza da d quadratica, decisamente migliore rispetto ai precedenti risultati, con le costanti che non dipendono da d.

L'ultimo risultato analizzato, quello di Seidel, porta un'idea molto semplice e intuitiva che arriva ad un risultato comunque buono, pari a $O(d!n)$, facendo sempre uso della componente randomica, ma sfruttando un'intuizione più geometrica rispetto a Clarkson, fornendo quello che risulta di gran lunga il più semplice dei tre algoritmi qui analizzati.

L'unico dei tre autori a presentare effettivamente un codice per il proprio algoritmo

è Clarkson, mentre per gli altri due saranno presentate solamente le idee che ci stanno alla base.

Per "Algoritmi di Las Vegas" si intende una famiglia di algoritmi aleatori che raggiungono l'ottimo di un problema (di un PL nel nostro caso) in un tempo non noto a priori, ma del quale può essere dato un valore atteso. Saranno certamente più noti al lettore degli algoritmi che sfruttano il cosiddetto "Metodo Montecarlo", che è in un certo senso speculare alla tipologia di algoritmi qui presentati e che sarà quindi utile per capire meglio cosa sono gli "Algoritmi di Las Vegas": l'idea del Metodo Montecarlo è di raggiungere il valore desiderato non come valore esatto ma in media, tramite delle scelte aleatorie che permettono di velocizzare il tempo di calcolo spesso facendo uso di approssimazioni. Un esempio classico di uso del Metodo Montecarlo è il calcolo di integrali nell'ambito della statistica. È possibile applicare il Metodo Montecarlo anche nel contesto della Programmazione Lineare, tenendo conto che in questo caso verrà raggiunto in media non l'ottimo, ma un valore che si discosta arbitrariamente poco da esso; questo perché, essendo i PL dei problemi di massimizzazione o minimizzazione vincolata, per poter raggiungere in media l'ottimo ad ogni scarto positivo da questo (maggiore dell'ottimo) dovrebbe corrispondere uno scarto negativo (minore dell'ottimo). Entrambi questi scarti per poter essere contati nella "media" dovrebbero essere ammissibili per il PL, ma ciò comporterebbe l'esistenza di valori ammissibili sia maggiori che minori dell'ottimo che ne violerebbe quindi l'ottimalità.
Gli algoritmi di Las Vegas, al contrario, non compiono un numero noto di iterazioni che prevedono al loro interno una scelta aleatoria come il Metodo Montecarlo, ma l'aleatorietà, data da una scelta compiuta all'inizio e/o nel corso dell'algoritmo, risiede proprio nel numero di iterazioni che porteranno l'algoritmo a terminare, mentre il valore di output sarà esattamente il valore ottimo, non un valore arbitrariamente vicino né un valore atteso.
I due algoritmi che possono essere definiti "di Las Vegas" in questo testo sono quelli di Clarkson (che usa proprio questo termine per definire il proprio algoritmo [3]) e di Seidel, mentre quello di Megiddo è semplicemente un algoritmo deterministico, che quindi non fa uso della componente aleatoria. Come sarà visto nei capitoli 3 e 4, gli algoritmi lì trattati non avranno un numero fissato di iterazioni, ma proseguiranno finché una certa condizione non risulterà soddisfatta. Il tempo di esecuzione sarà dunque una variabile aleatoria e ne verrà calcolato il valore atteso.

Megiddo e Clarkson condividono la medesima intuizione come base dei rispettivi algoritmi: sfruttare il fatto che l'ottimo del PL, come tutte le soluzioni di base, è un vertice del poliedro/simplesso che determina la regione ammissibile del problema, ed è quindi determinato univocamente come intersezione di d iperpiani di delimitazione. Riuscendo a scovare chi sono tali iperpiani "speciali" tra gli n vincoli totali, l'ottimo potrebbe essere trovato semplicemente ponendo ad uguaglianza le equazioni di tali iperpiani, riducendo di gran lunga la complessità del problema.

L'idea successiva di Megiddo è quella di "testare" alcuni iperpiani per vedere se l'ottimo si trova su essi o individuare in quale dei due semispazi delimitati dall'iperpiano tale ottimo giace, così da scartare (grazie ad un'attenta scelta degli iperpiani da testare) un numero notevole di vincoli ad ogni iterazione, riducendo ricorsivamente il

problema ad uno sempre più semplice. Il test fa uso di un "oracolo", capace di dare l'informazione necessaria, il quale si rivelerà essere nient'altro che un insieme di al più tre programmi lineari più semplici che possono essere risolti in modo poco dispendioso.

Clarkson invece introduce a questo punto la componente aleatoria: viene scelto a random un sottoinsieme R di vincoli tra gli n totali e viene calcolato ricorsivamente l'ottimo del problema su tali vincoli, qualora tale ottimo non rispetti alcuni degli n vincoli iniziali, significa che non è ammissibile per il PL di partenza, ma anche che V (l'insieme dei vincoli violati) contiene uno dei d vincoli "speciali" che ci interessa individuare. L'algoritmo permette (con tecniche diverse in base alle diverse versioni presentate nel capitolo 3, che andranno poi combinate così da avere il tempo medio migliore) di individuare con una buona probabilità (in tutto il capitolo si parlerà di valore atteso del tempo di calcolo per quanto già spiegato nella digressione sugli Algoritmi di Las Vegas) l'insieme dei vincoli "speciali" che ci permetterà quindi di trovare l'ottimo del PL.

Infine Seidel parte dalla stessa intuizione dei due predecessori, quella che l'ottimo sia determinato da d vincoli posti ad uguaglianza, e fa anch'egli uso della componente aleatoria: viene scelto a random un vincolo che viene tolto all'insieme totale, successivamente si risolve il PL definito sul nuovo insieme di vincoli. Se l'ottimo del nuovo PL soddisfa il vincolo scartato, allora è anche l'ottimo del problema originale, se invece viola tale vincolo, significa che quello scartato è uno dei d vincoli "speciali" e l'ottimo appartiene all'iperpiano che lo delimita. Si potrà quindi procedere ricorsivamente cercando l'ottimo non più in tutto lo spazio $\mathbb{R}^d$ ma solo sull'iperpiano di delimitazione di tale vincolo, riducendo di 1 la dimensione del problema.

In tutti i capitoli sarà necessario in certi punti l'ipotesi di esistenza di un unico ottimo per il PL (sia esso un punto o un raggio che certifichi l'illimitatezza del problema), per la quale basterà dare una condizione canonica che identifichi univocamente un ottimo in caso vi siano più soluzioni ammissibili che massimizzano/minimizzano la funzione obbiettivo. Un'idea su come procedere verrà data ogni volta che questo fatto è richiesto, ma un'analisi più approfondita è presente nella sezione 3.2 del capitolo dedicato all'algoritmo di Clarkson.
I problemi analizzati saranno di massimizzazione o di minimizzazione, a seconda di quale delle due possibilità risulterà più funzionale a rendere più semplice la spiegazione. Il caso di PL inammissibili (quando la regione ammissibile è vuota) sarà escluso dall'algoritmo di Clarkson, e quindi dovrà essere trovata una soluzione ammissibile prima di poter procedere, mentre non crea problemi per gli algoritmi di Megiddo e Seidel, che sono in grado di restituire l'inammissibilità così come il raggio per l'illimitatezza o le coordinate dell'ottimo.

Nel capitolo conclusivo, si tireranno le somme sui tre metodi presentati, riflettendo sull'efficienza di quanto esposto e accennando alle tecniche più innovative proposte nei vari metodi.

Capitolo 2

Megiddo

2.1 Introduzione

Verrà presentato in questa sezione l'algoritmo di Megiddo, o meglio solo un'idea su come possa essere costruito. Si tratta del primo algoritmo ad essere creato tra quelli presentati in questo testo, nel quale ancora non è presente l'idea di introdurre una componente aleatoria per velocizzare in valore atteso il tempo di calcolo. Si trovano però altri capisaldi da cui Clarkson e poi Seidel trarranno ispirazione: in primis le ipotesi di lavoro, considerando sistemi di n vincoli in d variabili dove d è generalmente molto più piccolo rispetto a n, e in secondo luogo il fatto di ricercare l'ottimo scartando ricorsivamente un ampio numero di vincoli che non lo definiscono univocamente come intersezione degli iperpiani che li delimitano.

Verrà dimostrato che un PL in d variabili e n vincoli può essere risolto in tempo $O(n)$ se d è fissato, e tale risultato sarà poi esteso a d maggiorati da funzioni che crescono lentamente in n.

Chiameremo un algoritmo "genuinely polynomial" (GP) quando richiede un numero di operazioni che è polinomiale in termini di nd (la dimensione della matrice sottostante al PL) con notazioni come sopra.

L'articolo analizzato in questo capitolo rappresenta un'estensione a dimensioni maggiori di un risultato già pubblicato dallo stesso Megiddo sull'esistenza di algoritmi GP in caso di PL in al più due variabili [8]. Analizzeremo nelle prossime pagine la complessità di un PL in uno spazio fissato, ossia il tempo asintotico in termini di n, numero di vincoli nello spazio $\mathbb{R}^d$ con d fissato. Questo porterà ad una dipendenza lineare in n con coefficiente di proporzionalità che cresce molto rapidamente in d (questa crescita rapida sarà il fattore che inciterà l'uso di scelte aleatorie nei prossimi capitoli).

La costruzione di tale algoritmo GP per d fissati potrà essere estesa anche per PL in cui d cresce lentamente in n come ad esempio $d = O((\log n/\log\log n)^{\frac{1}{2}})$, per cui il bound GP sarà valido anche in quei casi.

Alcune applicazioni di PL in cui la dimensione è piccola sono ad esempio:

- SEPARABILITÀ LINEARE: dati n punti in $\mathbb{R}^d$ in due insiemi disgiunti, trovare un iperpiano che li separa (o se non esiste, tale che sia minimizzata la distanza o la distanza al quadrato dei due insiemi dall'iperpiano), applicazione utile in statistica e in pattern recognition;
- APPROSSIMAZIONE DI CHEBYSHEV: dati n punti in $\mathbb{R}^d$: $a_i = (a_{i1}, \ldots, a_{id})$, $i = 1, \ldots, n$ vogliamo trovare la funzione lineare

$$f(x_1, \ldots, x_{d-1}) = \sum_{j=1}^{d-1} \alpha_j x_j + \alpha_d$$

che minimizzi $max\{|\sum_{j=1}^{d-1} \alpha_i a_{ij} + \alpha_d - a_{id}| : i = 1, \ldots, n\}$ estendibile per risolvere problemi di minimizzazione quadratica convessa.

2.2 Panoramica

Per cercare relativamente ad n oggetti (i vincoli) in uno spazio d-dimensionale, risolveremo ricorsivamente due problemi con $\frac{n}{2}$ oggetti in uno spazio $(d-1)$-dimensionale e poi il problema sarà ridotto a uno con $\frac{n}{2}$ oggetti in un d-spazio.

Saremo interessati a risolvere:

$$\begin{aligned} \min \quad & \sum_{j=1}^{d} c_j x_j \\ \text{soggetto a} \quad & \sum_{j=1}^{d} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, n \end{aligned} \tag{2.1}$$

È noto che solo alcuni vincoli definiscono univocamente la soluzione ottima di un PL; se fossimo in grado di identificarli, ci basterebbe risolvere un sistema di equazioni per trovarla. L'idea del metodo è di ridurre ripetutamente il numero di vincoli fino a quando il sistema non sia risolvibile direttamente. Per rendere efficace questa intuizione, dovremo scartare un buon numero di vincoli ad ogni round dell'algoritmo. Fortunatamente vedremo che esiste un modo di scartare una porzione fissata di vincoli indipendentemente da quanti essi sono.

Per semplicità di presentazione riscriviamo il PL di prima nella forma:

$$\begin{aligned} \min \quad & x_d \\ \text{soggetto a} \quad & x_d \geq \sum_{j=1}^{d-1} a_{ij} x_j + b_i, \quad i \in I_1 \\ & x_d \leq \sum_{j=1}^{d-1} a_{ij} x_j + b_i, \quad i \in I_2 \\ & 0 \geq \sum_{j=1}^{d-1} a_{ij} x_j + b_i, \quad i \in I_3 \end{aligned} \tag{2.2}$$

con $|I_1| + |I_2| + |I_3| = n$. La trasformazione è operata ad esempio chiamando $\tilde{x} = \sum_{j=1}^{d} c_j x_j$, eliminando x_d e rinominando $\tilde{x}$ come x_d. La modifica operata non influenza il comportamento dei vincoli in un sottospazio ortogonale al gradiente della funzione obbiettivo. Effettuare più volte un cambio del genere può risultare influente sul costo computazionale, ma ciò non è previsto dall'algoritmo ed è riportato solo per poter rendere più semplice la lettura di quanto segue. La forma a cui arriviamo dunque è:

$$\begin{aligned} \min \quad & x_d \\ \text{soggetto a} \quad & x_d \geq \max\left\{ \sum_{j=1}^{d-1} a_{ij}x_j + b_i : \quad i \in I_1 \right\} \\ & x_d \leq \min\left\{ \sum_{j=1}^{d-1} a_{ij}x_j + b_i : \quad i \in I_2 \right\} \\ & 0 \geq \max\left\{ \sum_{j=1}^{d-1} a_{ij}x_j + b_i : \quad i \in I_3 \right\} \end{aligned} \tag{2.3}$$

Si considerino due disuguaglianze con indici i,k nello stesso insieme, ad esempio in I_1:

$$\begin{aligned} x_d &\geq \sum_{j=1}^{d-1} a_{ij}x_j + b_i \\ x_d &\geq \sum_{j=1}^{d-1} a_{kj}x_j + b_k \end{aligned} \tag{2.4}$$

Il rapporto tra i due vincoli è uno dei seguenti:

- se $(a_{i1}, \ldots, a_{id-1}) = (a_{k1}, \ldots, a_{kd-1})$, uno dei due vincoli è ridondante (a seconda che $b_i > b_k$ o viceversa);
- sennò $\sum_{j=1}^{d-1} a_{ij}x_j + b_i = \sum_{j=1}^{d-1} a_{kj}x_j + b_k$ descrive un iperpiano di $\mathbb{R}^{d-1}$ che divide lo spazio in due domini: uno in cui $\sum_{j=1}^{d-1} a_{ij}x_j + b_i < \sum_{j=1}^{d-1} a_{kj}x_j + b_k$ e uno in cui vale "$>$". Se sapessimo da quale parte di questo iperpiano c'è l'ottimo potremmo sbarazzarci di uno dei due vincoli (dovendo essere $x_d \geq \sum_{j=1}^{d-1} a_{ij}x_j + b_i$ e $x_d \geq \sum_{j=1}^{d-1} a_{kj}x_j + b_k$ basta tenere il maggiore dei due).

Testare un iperpiano non è facile, per cui non testeremo mai un iperpiano solo per sbarazzarci di un unico vincolo. Testeremo relativamente pochi iperpiani, scelti con un metodo che permetterà di sbarazzarci di molti vincoli assieme. Vedremo un metodo di selezione degli iperpiani da testare, il cui risultato ci farà conoscere l'esito del test su molti altri iperpiani senza doverlo effettivamente compiere.

Dati n vincoli in d variabili, testeremo un numero costante di iperpiani (rispetto ad n, ma dipendente da d, una quantità che chiameremo $A(d)$) riguardo alla posizione di tali iperpiani rispetto l'ottimo. Un solo test corrisponde a risolvere fino a tre PL $(d-1)$-dimensionali di n vincoli (si veda sezione 2.5). Con le $A(d)$ interrogazioni, otteniamo i risultati dei test per $B(d) \cdot n$ coppie di disuguaglianze ($0 < B(d) \leq \frac{1}{2}$, anch'esso dipendente da d ma costante rispetto ad n); sapremo dire quale vincolo della

coppia potrà essere scartato, riducendo il problema ad uno sempre di dimensione d ma con $(1-B(d))\cdot n$ vincoli, sul quale si può procedere ricorsivamente.

Nei prossimi paragrafi sarà spiegato cosa si intende per testare un iperpiano e verrà dato un algoritmo che lo faccia. Questi test verranno poi organizzati, così da creare un algoritmo completo. Nella sezione 2.3 sarà presentato il problema di ricerca multidimensionale che sarà sviluppato nella sezione 2.4, facendo uso di un "oracolo" in grado di dare delle informazioni cruciali per lo svolgimento dell'algoritmo. La sezione 2.5 sarà dedicata alla costruzione di tale oracolo come PL risolvibile, infine nella sezione 2.6 i vari risultati saranno assemblati, dimostrando la validità del costo dell'algoritmo già precedentemente enunciato.

2.3 Il problema di ricerca multidimensionale

Consideriamo un problema di ricerca uno-dimensionale: esiste un x^* reale ignoto e un "oracolo" che è capace di dirci, dato un qualsiasi altro reale x, se $x < x^*$, $x = x^*$ o $x > x^*$. Dati n numeri $x_1, \dots, x_n$ si vuole conoscere la posizione di ognuno relativamente a x^*. Quante domande (e quale sarà il costo computazionale di queste domande) dovrò sottoporre all'oracolo per avere tale informazione? Possiamo ordinare in $O(n\log n)$ operazioni la lista $[x_1, \dots, x_n]$ e fare ricerca binaria per trovare la posizione di x^* nell'insieme ordinato, cosa che comporterà $O(\log n)$ interrogazioni.

Una strategia migliore asintoticamente è data usando un algoritmo per la ricerca della mediana in tempo lineare ([1], [10]): trovata questa, si può chiedere all'oracolo la sua posizione rispetto a x^* e scartare circa metà degli x_i e procedere ricorsivamente (gli x_i vengono scartati confrontandoli singolarmente con la mediana già trovata, cosa che viene fatta in tempo lineare e senza dover fare ulteriori interrogazioni). Un'interrogazione fatta rispetto all'elemento giusto può quindi bastare a dare l'esito per metà insieme. Questo secondo metodo richiede $O(n)$ operazioni per trovare la mediana in un insieme di n, $n/2$, $n/4, \dots$ elementi più $O(\log n)$ interrogazioni.

Un approccio ancora migliore (soprattutto a livello pratico) consiste nello scegliere a random un x_i e testarlo, poi scartare tutti gli elementi nella parte sbagliata e scegliere un altro elemento, sempre in modo uniforme tra gli elementi che restano. Questo approccio comporta un valore atteso di $O(\log n)$ interrogazioni più $O(n)$ comparazioni, e può essere reso ancora più efficiente cercando la mediana di un insieme di 3 o 5 elementi scelti randomicamente.

Generalizziamo ora il problema 1-dimensionale a d maggiori: sia $x^* \in \mathbb{R}^d$ non noto e supponiamo esista un oracolo che per ogni $a \in \mathbb{R}^d$ e $b \in \mathbb{R}$ sappia dirci se $a^t x^* < b$, o $a^t x^* > b$, cioè ci dice la posizione di x^* relativamente ad un iperpiano di $\mathbb{R}^d$.

Supponiamo di avere n iperpiani e di voler sapere la posizione di x^* relativamente ad essi, la domanda è ancora quante interrogazioni all'oracolo devo fare e qual è il costo computazionale necessario a sapere la posizione di x^* relativamente a n iperpiani. La generalizzazione è molto più complicata del caso 1-dimensionale, principalmente perché in $\mathbb{R}^d$ non c'è un ordine totale. Sembrerà quindi sorprendente come lo stesso risultato visto nel caso 1-dimensionale si generalizzi a dimensioni maggiori: sarà mostrato che per ogni d esiste una costante $C = C(d)$ (costante rispetto ad n ma dipendente

da d) tale che sono sufficienti $C \cdot \log n$ interrogazioni per determinare la posizione di x^* rispetto agli n iperpiani.

Ci sarà naturalmente un ulteriore costo computazionale che risulterà essere lineare in n con coefficiente di proporzionalità dipendente da d, come sarà visto nella prossima sezione.

2.4 La soluzione

Il procedimento dato è ricorsivo. Mostreremo che per ogni dimensione d esistono costanti indipendenti da n: $A = A(d)$, $B = B(d)$ con $0 < B \leq \frac{1}{2}$ tali che A interrogazioni siano sufficienti a determinare la posizione di x^* relativamente ad almeno $B \cdot n$ tra gli n iperpiani dati in $\mathbb{R}^d$. Se ciò è vero potremo scartare $B \cdot n$ iperpiani e procedere ricorsivamente: ogni round di A interrogazioni riduce il numero di iperpiani di un fattore $1-B$, quindi dopo al più $\log_{\frac{1}{1-B}} n$ round conosceremo la posizione di x^* relativamente a tutti gli iperpiani. Infatti: al k-simo round ci saremo ridotti a $(1-B)^k n$ vincoli; il caso base in cui non devo più ridurre è quando sono rimasto con un unico vincolo, che potrà essere testato singolarmente senza influenzare l'andamento asintotico. Il numero di round $\bar{k}$ sarà dunque determinato da:

$$(1-B)^{\bar{k}} n = 1 \Rightarrow n = \left(\frac{1}{1-B}\right)^{\bar{k}} \Rightarrow exp(\log n) = exp\left(\bar{k} \log \frac{1}{1-B}\right) \Rightarrow \bar{k} = \frac{\log n}{\log \frac{1}{1-B}}$$

Sarà dunque necessario un numero di interrogazioni pari a $O(\log n)$ (perché $\log \frac{1}{1-B}$ è una costante rispetto n in quanto lo è B). Il restante costo computazionale è pari a

$$cn + c(1-B)n + c(1-B)^2 n + \cdots < cn \sum_{i=0}^{\infty} (1-B)^i = cn \frac{1}{1-(1-B)} = \frac{c}{B} n$$

quindi lineare in termini di n. La linearità in n (più precisamente nel numero di vincoli, che si traduce in $(1-B)^k n$ al k-simo passaggio) del costo ulteriore con costante c dipendente da d sarà mostrata più avanti sempre all'interno di questa sezione, e verrà discusso anche in che modo c dipende dalla dimensione d.
Le costanti A e B saranno ottenute ricorsivamente in base alla dimensione; sappiamo già che per $d = 1$ con un'interrogazione scartiamo metà insieme di iperpiani $\Rightarrow A(1) = 1$, $B(1) = \frac{1}{2}$.

Vediamo per $d \geq 2$:
Siano $H_i := \{x \in \mathbb{R}^d \mid a_i^t x = b_i\}$, $i = 1, \ldots, n$ gli n iperpiani (con $a_i = (a_{i1}, \ldots, a_{id})^t \neq 0$ e $b_i \in \mathbb{R}$ $\forall i = 1, \ldots, n$). Potrebbe essere conveniente fare in seguito un cambio di variabili per semplicità di spiegazione, tuttavia, non potendo operare il cambio su x^* ignoto, dovremo ritrasformare l'equazione dell'iperpiano nel sistema originale prima di consultare l'oracolo riguardo la posizione di tale iperpiano. Più precisamente, se un vettore $x \in \mathbb{R}^d$ è rappresentato come $x = My$, dove M è una matrice $d \times d$ non singolare, un iperpiano $a^t x = b$ è rappresentato da $(a^t M)y = b$ e possiamo lavorare con vettori $a' = M^t a$; se serviranno informazioni riguardo l'iperpiano $a'^t y = b$ guarderemo $a^t x = b$ con $a^t = a'^t M^{-1}$.

Vorremmo un sistema di coordinate in cui ogni iperpiano interseca il sottospazio $\langle x_1, x_2 \rangle$ in una retta, ossia un sistema per cui tale intersezione sia sempre non vuota e non sia mai tutto il sottospazio $\langle x_1, x_2 \rangle$. Dato che abbiamo un numero finito di iperpiani, un tale sistema esiste sempre, ed è trovato tramite trasformazioni non singolari. Nello specifico vorremmo avere $(a_{i1}, a_{i2}) \neq (0,0) \ \ \forall i = 1, \ldots, n$.

- È possibile trovare in tempo $O(n)$ una base per $\mathbb{R}^d$ relativamente alla quale si ha $a_{ij} \neq 0 \ \forall i, j$. Questo si basa sull'osservazione che un elemento $v = v(\varepsilon) = (1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{d-1})^t$ è ortogonale a degli a_i per al più $n(d-1)$ valori di ε (perché gli a_i sono in numero finito mentre gli ε variano con continuità). Potremo dunque scegliere una base di vettori così che nessuno sia ortogonale a nessun a_i.

- Alternativamente, per evitare troppe trasformazioni, possiamo semplicemente ignorare gli iperpiani con $a_{i1} = a_{i2} = 0$; se ce ne sono troppi per poter essere ignorati, basta semplicemente selezionare altre variabili al posto di x_1, x_2; se il problema sussiste su tutte le coppie, vuol dire che la maggior parte dei vincoli ha un unico $a_{ij} \neq 0$ e quindi il problema è estremamente semplice e non necessita di tutta questa trattazione.

Quindi ogni iperpiano H_i interseca il sottospazio $\langle x_1, x_2 \rangle$ in una retta $a_{i1}x_1 + a_{i2}x_2 = b_i$. Definiamo l'inclinazione di H_i come l'inclinazione di tale retta, ovvero $+\infty$ se $a_{i2} = 0$, $-\frac{a_{i1}}{a_{i2}}$ se $a_{i2} \neq 0$.

Vorremmo che almeno metà degli iperpiani avesse inclinazione non negativa e almeno metà l'avesse non positiva. Si può ottenere questa condizione tramite trasformazione lineare del sottospazio $\langle x_1, x_2 \rangle$, trovando la mediana dell'insieme delle inclinazioni e portandola a 0 tramite trasformazione lineare. Questa operazione richiede tempo lineare in n, ma è comunque necessaria solo ai fini della spiegazione; l'algoritmo può essere implementato per eseguire tale trasformazione, a noi basterà assumere che i coefficienti verifichino fin da subito la proprietà.

Il primo passo della procedura consiste nel creare coppie disgiunte di iperpiani tali che in ogni coppia almeno un elemento abbia inclinazione non negativa e almeno un elemento abbia inclinazione non positiva. Consideriamo ora la relazione tra due membri di una coppia: H_i di inclinazione non negativa e H_k di inclinazione non positiva, accoppiati durante il processo.

- Assumiamo che le equazioni di H_i, H_k siano linearmente indipendenti. Definiamo $H_{ik}^{(1)}$ l'iperpiano definito da

$$\sum_{j=1}^{d} (a_{k1}a_{ij} - a_{i1}a_{kj})x_j = a_{k1}b_i - a_{i1}b_k$$

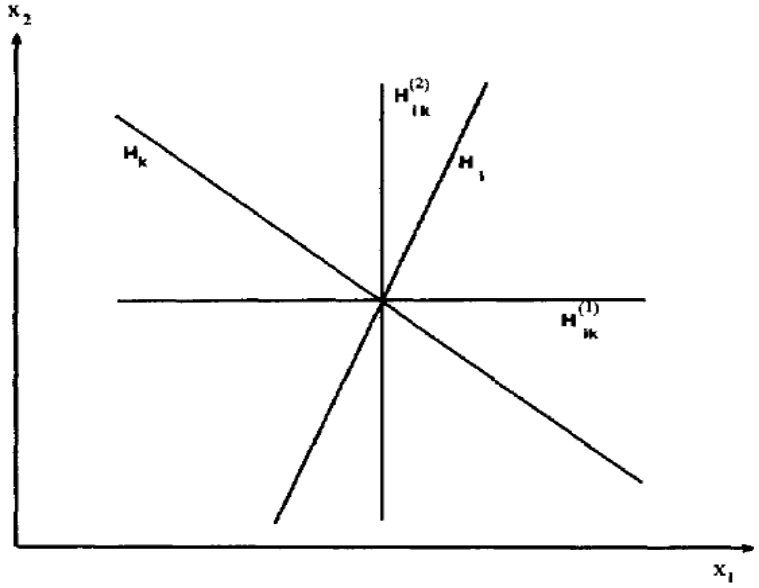
  ottenuta sottraendo a_{i1} volte l'equazione di H_k da a_{k1} volte l'equazione di H_i. Per come è definito, l'intersezione tra H_i, H_k e $H_{ik}^{(1)}$ è la stessa di quella tra i soli H_i, H_k, ed ha dunque dimensione $(d-2)$; inoltre il coefficiente di x_1 in $H_{ik}^{(1)}$ è $a_{k1}a_{i1} - a_{i1}a_{k1} = 0$. Questa seconda proprietà sarà essenziale per come tratteremo gli iperpiani della forma $H_{ik}^{(1)}$ in un contesto di dimensione minore (ossia sui sottospazi).

In maniera analoga sia $H_{ik}^{(2)}$ l'iperpiano definito da

$$\sum_{j=1}^{d}(a_{k2}a_{ij}-a_{i2}a_{kj})x_j = a_{k2}b_i - a_{i2}b_k$$

per il quale valgono caratteristiche simili a quelle di $H_{ik}^{(1)}$ (qui però è il coefficiente di x_2 ad essere pari a 0).

Osserviamo che, nota la posizione di x^* rispetto ad entrambi $H_{ik}^{(1)}$ e $H_{ik}^{(2)}$, possiamo già stabilire la posizione di x^* rispetto ad uno tra H_i e H_k. Per vedere ciò si noti che l'intersezione tra i 4 iperpiani è ancora $(d-2)$-dimensionale e che le posizioni relative sul sottospazio $\langle x_1, x_2\rangle$ danno una caratterizzazione completa delle posizioni relative su tutto $\mathbb{R}^d$ (sono sostanzialmente proiettate). La pendenza definita sopra è come la pendenza usuale se interpretiamo x_1 come x e x_2 come y. Dunque avremo che $H_{ik}^{(1)}$ avendo coefficiente di x_1 nullo sarà orizzontale su tale piano e allo stesso modo $H_{ik}^{(2)}$ sarà verticale.



Sapere che x^* si trova a destra di $H_{ik}^{(2)}$ e in basso rispetto a $H_{ik}^{(1)}$ permette di concludere che x^* sta nel "quarto quadrante" delle intersezioni tra $H_{ik}^{(1)}$ e $H_{ik}^{(2)}$ e si può quindi concludere che si trova in basso a destra rispetto a H_i (aiutarsi con la figura per visualizzare). Allo stesso modo se è noto che x^* è a destra di $H_{ik}^{(2)}$ e in alto rispetto a $H_{ik}^{(1)}$ si può già dire che x^* è in alto a destra rispetto H_k, e così anche gli altri casi. (Si noti che è assolutamente necessario che le due inclinazioni siano opposte in segno per poter fare questo tipo di ragionamento.)

Formalizziamo quest'idea: supponiamo che H_i abbia inclinazione positiva e che $a_{i1} < 0 < a_{i2}$ (perché l'inclinazione sia positiva i due coefficienti devono avere

segni opposti, l'altra possibilità è analoga). Le condizioni per cui x^* giace da un lato di $H_{ik}^{(1)}$ e $H_{ik}^{(2)}$ sono note per ipotesi (gli altri casi si svolgono allo stesso modo):

$$\sum_{j=1}^{d}(a_{k1}a_{ij}-a_{i1}a_{kj})x_j^* < a_{k1}b_i - a_{i1}b_k; \qquad \sum_{j=1}^{d}(a_{k2}a_{ij}-a_{i2}a_{kj})x_j^* < a_{k2}b_i - a_{i2}b_k$$

Moltiplicando la prima per a_{i2} e la seconda per a_{i1}, e dopo aver sottratto la seconda appena ottenuta alla prima ottengo:

$$\sum_{j=1}^{d}[(a_{k1}a_{ij}-a_{i1}a_{kj})a_{i2}-(a_{k2}a_{ij}-a_{i2}a_{kj})a_{i1}]x_j^* < a_{i2}a_{k1}b_i - a_{i2}a_{i1}b_k - a_{i1}a_{k2}b_i + a_{i1}a_{i2}b_k$$

$$\sum_{j=1}^{d}[a_{k1}a_{ij}a_{i2}-a_{i1}a_{kj}a_{i2}-a_{k2}a_{ij}a_{i1}+a_{i2}a_{kj}a_{i1}]x_j^* < a_{i2}a_{k1}b_i - a_{i1}a_{k2}b_i$$

$$\sum_{j=1}^{d}[a_{k1}a_{ij}a_{i2}-a_{k2}a_{ij}a_{i1}]x_j^* < a_{i2}a_{k1}b_i - a_{i1}a_{k2}b_i$$

$$\sum_{j=1}^{d}[a_{k1}a_{i2}-a_{k2}a_{i1}]a_{ij}x_j^* < [a_{i2}a_{k1}-a_{i1}a_{k2}]b_i$$

$$[a_{k1}a_{i2}-a_{k2}a_{i1}]\sum_{j=1}^{d}a_{ij}x_j^* < [a_{i2}a_{k1}-a_{i1}a_{k2}]b_i$$

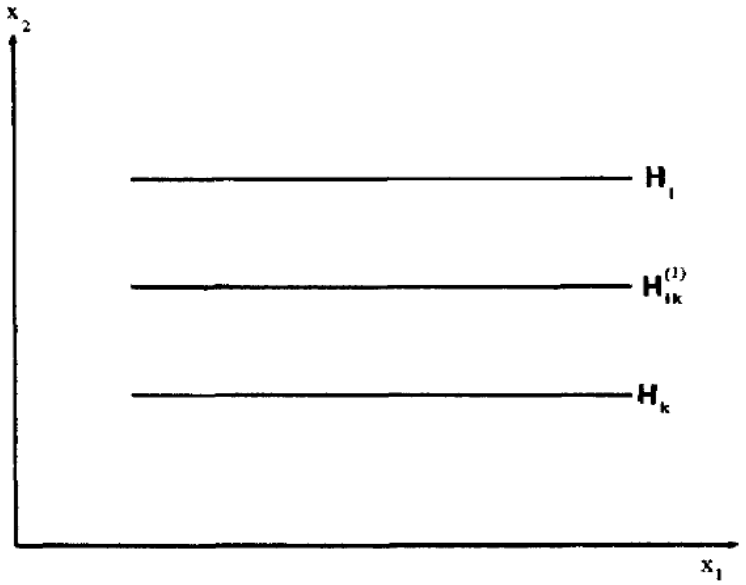
Che dà, a seconda del segno di $a_{i2}a_{k1}-a_{i1}a_{k2} \neq 0$ per indipendenza,

$$\sum_{j=1}^{d}a_{ij}x_j^* < b_i \quad \text{oppure} \quad \sum_{j=1}^{d}a_{ij}x_j^* > b_i$$

Che ci dice qual è la posizione di x^* rispetto H_i (in questo esempio).

- Se H_i e H_k sono linearmente dipendenti, si ha $a_{i1} = a_{k1} = 0$: questo segue dal fatto che H_i e H_k hanno la stessa inclinazione in quanto linearmente dipendenti e questa deve essere non positiva e non negativa. $H_{ik}^{(1)}$ viene definito come l'iperpiano parallelo a H_i e H_k che giace a metà fra essi. Formalmente: la dipendenza lineare si traduce nell'esistenza di una costante reale $\lambda \neq 0$ tale che $a_{ij} = \lambda a_{kj}$. L'equazione che definisce $H_{ik}^{(1)}$ è quindi:

$$\sum_{j=1}^{d}a_{ij}x_j = \frac{1}{2}(b_i + \lambda b_k)$$



Ovviamente $H_{ik}^{(1)}$ ha coefficiente di $x_1 = 0$ anche in questo caso (somma di coefficienti nulli). Chiaramente, nota la posizione di x^* rispetto $H_{ik}^{(1)}$, è nota quella rispetto uno tra H_i e H_k: se è sopra $H_{ik}^{(1)}$ è anche sopra H_k; se è sotto $H_{ik}^{(1)}$, è anche sotto H_i. È possibile una trattazione formale anche di questo caso, pressoché analoga alla precedente, che viene omessa.

Consideriamo ora gli iperpiani $H_{ik}^{(1)}$ per ogni coppia (i,k) che è stata composta. Dato che questi hanno tutti coefficiente di x_1 pari a 0, le loro intersezioni col sottospazio $(d-1)$-dimensionale $\langle x_2,\ldots,x_d\rangle$ hanno tutte dimensione $(d-2)$ e possono essere quindi intese come iperpiani su tale sottospazio. Possiamo applicare ora alla collezione di tali iperpiani il processo di ricerca che è noto ricorsivamente per iperpiani in $\mathbb{R}^{d-1}$. L'oracolo che abbiamo per x^* in $\mathbb{R}^d$ può essere usato come oracolo per la proiezione di x^* su $\langle x_2,\ldots,x_d\rangle$: dato che $H_{ik}^{(1)}$ ha coefficiente di $x_1 = 0$, segue che la proiezione di x^* su $\langle x_2,\ldots,x_d\rangle$ giace da una dalle due parti dell'intersezione di $H_{ik}^{(1)}$ con $\langle x_2,\ldots,x_d\rangle$ se e solo se x^* giace dal lato corrispondente di $H_{ik}^{(1)}$ su tutto $\mathbb{R}^d$. (Stiamo espandendo un concetto già usato nel caso 2-dimensionale del sottospazio $\langle x_1,x_2\rangle$, che avevamo chiamato intuitivo perché lavorava solo con punti e rette).

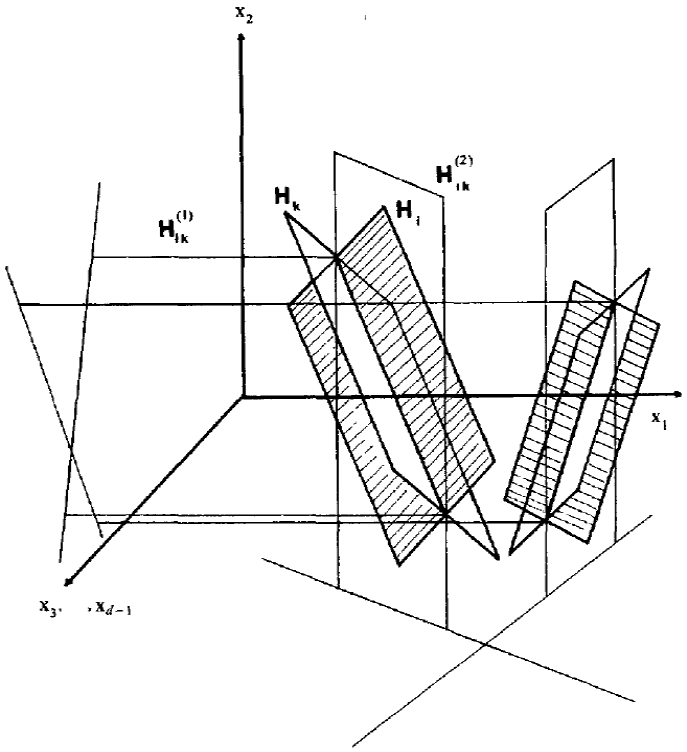
Un ragionamento del tutto analogo può essere fatto per gli iperpiani del tipo $H_{ik}^{(2)}$ nel sottospazio $\langle x_1,\, x_3,\ldots,x_d\rangle$.

Le operazioni fatte fino a questo punto sono state:

- La creazione di coppie di iperpiani tali che uno abbia inclinazione non negativa e uno non positiva, chiaramente realizzato con costo lineare nel numero di iperpiani n;

- La creazione degli $H_{ik}^{(1)}$, $H_{ik}^{(2)}$ (i secondi solo nel caso di indipendenza lineare tra i rispettivi H_i e H_k) per ognuna delle coppie: ogni creazione prevede una somma di d termini e al più n di tali iperpiani devono essere formati, per un totale ancora lineare in n;
- Le interrogazioni rivolte all'oracolo riguardanti gli $H_{ik}^{(1)}$, $H_{ik}^{(2)}$ (o solo parte di questi come si vedrà nei due diversi approcci presentati tra poco) avranno un costo computazionale che non riguarderà il "costo ulteriore" ma andrà compreso nel costo che avevamo chiamato $A(d)$;
- Infine i confronti: per le coppie di cui conoscerò le posizioni rispetto entrambi i $H_{ik}^{(1)}$, $H_{ik}^{(2)}$ dovrò applicare dei confronti per vedere se tali informazioni mi permetteranno di conoscere la posizione rispetto l'iperpiano H_i o rispetto H_k: si tratta di due operazioni di confronto per ognuna di queste coppie, che sono una frazione del totale di quelle formate: $\frac{n}{2}$; un costo che risulta anche in questo caso lineare in n.

Ciò prova la linearità del costo ulteriore rispetto al numero totale di vincoli, con costante c: essa è lineare in d e quindi questa dipendenza è ininfluente visto che la $B(d)$ avrà una dipendenza maggiore.



Vi sono ora due diversi approcci per applicare la ricorsione:

1. Interrogando l'oracolo $A(d-1)$ volte (ricordiamo che la parentesi indica la dipendenza di A da $d-1$) otteniamo informazioni di x^* relativamente a $B(d-1)\cdot\frac{n}{2}$

iperpiani della forma $H_{ik}^{(1)}$ dove (i,k) è una delle coppie precedentemente formate. ($B(d-1)$ moltiplica $\frac{n}{2}$ perché questo è il numero degli iperpiani del tipo $H_{ik}^{(1)}$). Guardiamo ora gli iperpiani della forma $H_{ik}^{(2)}$, ma solo per le coppie (i,k) per cui conosciamo la posizione di x^* relativamente a $H_{ik}^{(1)}$. Con altre $A(d-1)$ interrogazioni, ci saranno $(B(d-1))^2\frac{n}{2}$ coppie di iperpiani H_i, H_k per cui conosciamo la posizione di x^* relativamente ad uno dei due membri, grazie a quanto visto in precedenza, dato che è nota la posizione di x^* rispetto ad entrambi $H_{ik}^{(1)}$ e $H_{ik}^{(2)}$.

Dunque con $2A(d-1)$ interrogazioni conosciamo la posizione di x^* rispetto a una frazione pari a $\frac{1}{2}(B(d-1))^2$ del numero di iperpiani. Definiamo dunque $A(d) = 2A(d-1)$; $B(d) = \frac{1}{2}(B(d-1))^2$. Questo, unito a quanto già detto per i casi base $A(1) = 1$; $B(1) = \frac{1}{2}$ ci dà immediatamente la soluzione alle equazioni ricorsive pari a:

$$A(d) = 2^{d-1} \qquad B(d) = 2^{1-2^d}$$

Il numero di interrogazioni totali sarà dato dal numero di interrogazioni per round $A(d)$ per il numero di round, che avevamo già visto essere in generale pari a $\log_{\frac{1}{1-B}} n$, nello specifico il numero di interrogazioni all'oracolo sarà $C(d)\cdot\log n$ con $C(d)$ dell'ordine di $2^{O(2^d)}$:

$$(\log_{\frac{1}{1-B(d)}} n)\cdot A(d) = \frac{\log n}{-\log(1-B(d))}\cdot A(d) = -\frac{\log n}{\log(1-2^{1-2^d})}\cdot 2^{d-1} = \log n\cdot C(d)$$

$$C(d) = -2^{d-1}\frac{1}{\log(1-2^{1-2^d})} = -2^{d-1}\frac{1}{-2^{1-2^d}-\ldots} <$$

$$< 2^{d-1}\frac{1}{2^{1-2^d}} = 2^{d-1}\cdot 2^{2^d-1} = 2^{2^d+d-2} = 2^{O(2^d)}$$

dove si sono usati lo sviluppo di Taylor $\log(1-x) = -x-\frac{x^2}{2}-\ldots$, per x vicino a 0, pari a 2^{1-2^d} nel nostro caso, che è inoltre sempre positivo e ci permette dunque di maggiorare la serie di Taylor a denominatore con solo il primo termine come fatto nel terzo passaggio del calcolo di $C(d)$.

Si noti inoltre che nel caso $d = 1$ si ha $C(1) = \frac{1}{\log 2}$ che può essere anche preso $C(1) = 1$ senza creare problemi a livello asintotico. Questo sarà utile come caso base quando verranno tratte le conclusioni nella sezione 2.6.

Un ulteriore costo è necessario e pari a $O(F(d)\cdot n)$ con $F(d)$ anch'esso dell'ordine di $2^{O(2^d)}$:

$$F(d) = \frac{c}{B} = \frac{c}{2^{1-2^d}} = c\cdot 2^{2^d-1}$$

2. Supponiamo di scoprire ricorsivamente la posizione di x^* rispetto a tutti gli iperpiani $H_{ik}^{(1)}, H_{ik}^{(2)}$ (per ogni coppia formata (i,k)). Definiamo $Q(n,d)$ il numero di interrogazioni necessarie per n iperpiani in $\mathbb{R}^d$ e $T(n,d)$ il costo computazionale aggiuntivo dello stesso problema. Risolvendo due problemi $(d-1)$-dimensionali

con $\frac{n}{2}$ iperpiani (per $H_{ik}^{(1)}$ e $H_{ik}^{(2)}$ rispettivamente, ricordando che possono essere interpretati come iperpiani di $\mathbb{R}^{d-1}$ per via dei coefficienti di x_1, x_2 uguali a 0) sarà nota la posizione di metà degli n iperpiani di partenza: uno per ogni coppia. Chiaramente $Q(n,d) \leq n$ perché posso interrogare l'oracolo riguardo ogni iperpiano, indipendentemente dalla dimensione del problema; si può quindi dare la ricorrenza:

$$Q(n,d) = \min\left\{n,\ 2Q\left(\frac{n}{2},d-1\right)+Q\left(\frac{n}{2},d\right)\right\}$$

dove il $Q(n/2,d)$ è necessario a trovare la posizione dei rimanenti vincoli. I casi base sono $Q(n,1) = 1+\lfloor \log_2 n \rfloor$ (usando la strategia migliore a livello asintotico trattata nella sezione 2.3) e $Q(1,d) = 1$. Sarà più semplice lavorare senza il minimo, ossia risolvere

$$\widetilde{Q}(n,d) = 2\widetilde{Q}\left(\frac{n}{2},d-1\right)+\widetilde{Q}\left(\frac{n}{2},d\right)$$

con stessi casi base e poi usare $Q(n,d) \leq \min\{n,\ \widetilde{Q}(n,d)\}$ dove in generale NON vale l'uguaglianza (ad esempio $Q(32,2) = 26$ mentre $\widetilde{Q}(32,2) = 31$, esempio proposto dallo stesso autore).

Considero $n = 2^L$, con L intero; posso ricondurmi a questo caso poiché mi accorgo che il caso peggiore, che è quello che sono interessato a trattare, avviene proprio quando n è una potenza di 2; posso considerare invece di $\widetilde{Q}$ una ricorrenza del tipo $F(L,d) = 2F(L-1,d-1)+F(L-1,d)$ con casi base $F(L,1) = L+1$, $F(0,d) = 1$, la cui soluzione è trovata come in [9] (a sua volta da [2])

$$F(L,d) = 2^d \sum_{i=1}^{L+2-d} i \binom{L-i}{d-2} + \sum_{j=0}^{d-2} \binom{L}{j} 2^j \quad \Rightarrow \quad F(L,d) < \frac{(2L)^d}{(d-2)!}$$

Dimostrazione. Questa implicazione è giustificata trovando opportune maggiorazioni per le due sommatorie, per la prima:

$$2^d \sum_{i=1}^{L+2-d} i \binom{L-i}{d-2} \leq 2^d (L+2-d)^2 \binom{L-1}{d-2} \leq 2^d (L+2-d)^2 \frac{L^{d-2}}{(d-2)!}$$

dove nella prima disuguaglianza si maggiora i con il massimo possibile cioè $(L+2-d)$ (nell'ipotesi che $d << L$ che sarà mantenuta per il resto della dimostrazione) e il coefficiente binomiale con il massimo possibile, per $i = 1$, e sommo il massimo per il numero totale di addendi, ancora $(L+2-d)$; nella seconda uso Stirling e maggioro $L-1 < L$. Per la seconda sommatoria:

$$\sum_{j=0}^{d-2} \binom{L}{j} 2^j \leq (d-1)\binom{L}{d-2} 2^{d-2} \leq (d-1)\frac{L^{d-2}}{(d-2)!} 2^{d-2}$$

dove nella prima disuguaglianza maggioro la parte sommata con il maggiore dei valori assunti al variare di j, ottenuto per $j = d-2$ ($d << L$), mentre nella secondo uso Stirling.

Ora sommando le due maggiorazioni:

$$2^d(L+2-d)^2\,\frac{L^{d-2}}{(d-2)!}+(d-1)\frac{L^{d-2}}{(d-2)!}2^{d-2}=\frac{L^{d-2}}{(d-2)!}2^{d-2}[4(L+2-d)^2+(d-1)]$$

la quantità tra parentesi quadre può tranquillamente essere approssimata da $4L^2$ perché $L >> d$ per cui la somma in totale è maggiorata da

$$\frac{L^{d-2}}{(d-2)!}2^{d-2}4L^2=\frac{L^d}{(d-2)!}2^d=\frac{(2L)^d}{(d-2)!}$$

□

Questo bound per d fissato dà

$$Q(n,d)\leq \min\left\{n,\ \frac{(2\log_2\ n)^d}{(d-2)!}\right\}\leq(\log_2 n)^d\,\frac{2^d}{(d-2)!}=O((\log_2 n)^d)$$

con costante $C=C(d)=\frac{2^d}{(d-2)!}$.

Per quanto riguarda il costo ulteriore T si ha:

$$T(n,d)\leq 2T\left(\frac{n}{2},d-1\right)+T\left(\frac{n}{2},d\right)+\theta(nd)$$

Dove $\theta(nd)$ indica il numero di operazioni extra svolte ed è lineare in n e in d (in n per quanto già visto e in d perché certe operazioni vanno ripetute per un numero pari alla dimensione del problema). Considerando la ricorrenza $F(L,d)=2F(L-1,d-1)+F(L-1,d)+2^Ld$ di casi base $F(L,1)=2^L$, $F(0,d)=1$. Questa risulta lineare in 2^L per ogni d fissato, con costante $C=C(d)$ che soddisfa $C(d)\leq 2(C(d-1)+d)$

Dimostrazione. Per induzione doppia su L e d: dimostrati validi i casi in $(L-1,d-1)$ e $(L-1,d)$ voglio mostrare la tesi in (L,d); i casi base sono validi per le coppie $(0,d)$ e $(L,1)$ per ogni d e per ogni L. Per visualizzare il meccanismo di induzione si pensi ad un reticolo su $\mathbb{R}^2$ isomorfo a $\mathbb{N}^2$ in cui L sta in ascissa e d in ordinata. Assumo di risolvere l'induzione "per righe": ossia fissando una d e dimostrando i casi per ogni (L,d) a quel d fissato.

Grazie a questo meccanismo ho che per dimostrare (L,d), i casi $(L-1,d-1)$ e $(L-1,d)$ sono effettivamente noti: valga dunque $F(L-1,d-1)\leq C(d-1)2^{L-1}$ e $F(L-1,d)\leq C(d)2^{L-1}$, allora

$$F(L,d)=2F(L-1,d-1)+F(L-1,d)+2^Ld\leq 2C(d-1)2^{L-1}+C(d)2^{L-1}+2^Ld=$$

$$=2^L\left(C(d-1)+d+\frac{C(d)}{2}\right)=:C(d)2^L$$

Si ha quindi $C(d)\leq C(d-1)+d+\frac{C(d)}{2}\Rightarrow\frac{C(d)}{2}\leq C(d-1)+d\Rightarrow C(d)\leq 2(C(d-1)+d)$ □

Da questo risultato consegue che $T(n,d) < d2^d n$, ottenuto tornando a sostituire $n = 2^L$ e osservando che $C(d) \leq 2(C(d-1)+d)$ è risolto addirittura ad uguaglianza da $C(d) = 2^{d+2} - 2d - 4$: $C(1) = 2 \leq 2$ e

$$2(C(d-1)+d) = 2(2^{(d-1)+2} - 2(d-1) - 4 + d) = 2(2^{d+1} - 2d + 2 - 4 + d) =$$

$$= 2(2^{d+1} - d - 2) = 2^{d+2} - 2d - 4 = C(d)$$

Si noti come le ultime righe danno un risultato migliore di quello enunciato su $T(n,d)$: $T(n,d) \leq 2^{d+2} n$ (migliore si intende nella condizione che $d > 4$). Continueremo tuttavia ad usare il bound $T(n,d) < d2^d n$, che è quello utilizzato dall'autore, che sarà sufficiente ai fini del risultato da mostrare.

Il secondo approccio risulta migliore rispetto al primo, se n non è estremamente grande rispetto a d (per una spiegazione migliore si veda al termine della sezione 2.6). I due approcci di ricerca multidimensionale portano a due diversi algoritmi, che saranno discussi dopo aver parlato dell'oracolo per la programmazione lineare.

2.5 L'oracolo per la Programmazione Lineare

Vogliamo specificare in questa sezione cosa si intende per "testare" un iperpiano. Dato un PL in d variabili

$$\begin{aligned} &\min \quad \sum_{j=1}^{d} c_j x_j \\ &\text{soggetto a} \quad \sum_{j=1}^{d} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, n \end{aligned} \tag{2.5}$$

ci serve una procedura che dia un ottimo, certifichi l'illimitatezza o, in caso di inammissibilità, dia un vettore $x' = (x'_1, \ldots, x'_d)$ che minimizzi la funzione

$$f(x_1, \ldots, x_d) = \max \left\{ b_i - \sum_{j=1}^{d} a_{ij} x_j \ : \ i = 1, \ldots, n \right\}$$

L'inammissibilità del problema corrisponde al fatto che per ogni $x \in \mathbb{R}^d$ esiste almeno una coordinata i per cui $b_i > \sum a_{ij} x_j$ e quindi per ogni x la funzione f sarà > 0 e tale sarà quindi anche il minimo (l'esistenza di tale minimo può essere mostrata applicando il teorema di Weierstrass alla funzione f, chiaramente continua, dopo essersi ridotti ad un opportuno insieme compatto).

La richiesta nel caso dell'inammissibilità, per quanto presto spiegata, può sembrare inusuale, ma risulterà fondamentale nella ricorsione che vedremo nel CASO 2 (per verificare l'impossibilità che due casi avvengano contemporaneamente).

Il processo descritto fino alla sezione precedente permette di risolvere un PL come quello appena scritto a patto di avere un oracolo adatto. Nello specifico, dato un iperpiano $\sum_{j=1}^{d} a_{ij} x_j = b_i$ dobbiamo sapere:

1. se l'iperpiano contiene il risultato: ossia se l'ottimo vi appartiene, o se il PL è illimitato anche sull'iperpiano o se un x' che minimizza f giace sull'iperpiano;

2. se il risultato si trova da una certa parte dell'iperpiano; in questo caso non ci aspettiamo di conoscere la natura di tale risultato, che dipende dalla limitatezza e dall'ammissibilità del problema.

Tale approccio è valido anche se il PL ammette più ottimi: è noto che l'insieme delle soluzioni ottime è convesso, per cui se l'iperpiano non contiene un ottimo, allora tutti gli ottimi saranno dalla stessa parte dell'iperpiano e l'oracolo darà la stessa risposta indipendentemente da quale ottimo è scelto come x^*. Mostreremo ora che l'oracolo non è altro che un'applicazione ricorsiva del Master Problem (MP), ossia il PL (2.5), con quella accezione un po' strana per certificare l'inammissibilità, applicato in spazi di dimensione minore.

Dato il MP e l'iperpiano da valutare, consideriamo come prima cosa lo stesso problema con l'equazione dell'iperpiano che viene aggiunta come vincolo: otteniamo un PL in d variabili e $n+1$ vincoli, che può essere ridotto a un PL di $d-1$ variabili e n vincoli eliminando una variabile tramite l'equazione dell'iperpiano. Per semplicità assumiamo che l'iperpiano sia $x_d = 0$, sennò possiamo semplicemente applicare una trasformazione affine che lo renda tale. Il problema $d-1$ dimensionale è quindi ottenuto da quello originale semplicemente scartando l'ultima variabile x_d. Assumiamo $d \geq 2$, poiché il caso $d = 1$ è banale dato che l'iperpiano è in realtà un punto.

- Se il problema è ammissibile e illimitato su $\{x_d = 0\}$, lo sarà anche il problema originale e si può concludere;

- sennò otteniamo ricorsivamente l'ottimo $x^* = (x_1^*, \ldots, x_{d-1}^*, 0)$ relativo a $\{x_d = 0\}$, o un vettore $x' = (x_1', \ldots, x_{d-1}', 0)$ che minimizza f su $\{x_d = 0\}$ (a seconda che il nuovo PL ridotto sia ammissibile o no.)

Dobbiamo determinare ora se abbiamo già il risultato, cioè se quanto trovato come risultato su $\{x_d = 0\}$ vale anche su tutto lo spazio $\mathbb{R}^d$, oppure da che parte del vincolo $\{x_d = 0\}$ si trova il risultato su $\mathbb{R}^d$. Distinguiamo due casi:

- CASO 1: supponiamo di aver trovato ricorsivamente un ottimo x^* di $\{x_d = 0\}$. Vogliamo capire ora se esiste un vettore $y = (y_1, \ldots, y_d)$ con $y_d > 0$, $\sum c_j y_j < \sum c_j x_j^*$ (il MP è un problema di minimizzazione) e $\sum a_{ij} y_j \geq b_i \ \ , \forall i = 1, \ldots, n$, cioè se esiste un "nuovo ottimo" del MP, cioè senza vincolo aggiuntivo $\{x_d = 0\}$ (in particolare si chiede se esiste nella regione $\{x_d > 0\}$). Dato che la regione ammissibile di un PL è convessa, e grazie alla linearità della funzione obbiettivo, se un tale y esiste, allora soddisferanno le stesse richieste anche i vettori della forma $(1-\varepsilon)x^* + \varepsilon y$, $\forall\, 0 < \varepsilon \leq 1$. Basterà dunque cercare in un intorno di x^*. Nello specifico si consideri il seguente problema $(d-1)$-dimensionale:

$$\begin{aligned} \min \quad & \sum_{j=1}^{d-1} c_j x_j \\ \text{soggetto a} \quad & \sum_{j=1}^{d-1} a_{ij} x_j + a_{id} \geq 0, \quad i \in I \end{aligned} \tag{2.6}$$

con $I = \{i \ : \sum a_{ij} x_j^* = b_i\}$. Vengono considerati solo i vincoli rispettati ad uguaglianza da x^* (ossia in vincoli in cui non c'è libertà di movimento, neanche infinitesimo, per non rischiare di uscire dalla regione ammissibile) e viene aggiunto implicitamente il vincolo $x_d = 1$; questo perché siamo interessati al comportamento locale nell'intorno di x^* e basta determinare l'esistenza di una direzione in cui la soluzione migliora (si noti che la direzione scelta ha $x_d = 1$, concordemente alla richiesta $y_d > 0$ fatta in precedenza: si sta cercando di migliorare nella regione $\{x_d > 0\}$). È facile vedere che questo problema $(d-1)$-dimensionale ha soluzione con valore della funzione obbiettivo negativo se e solo se esiste una direzione nel semispazio $\{x_d > 0\}$ tale che muovendo di un ε (piccolo ma positivo) la x^* verso tale direzione, si resta nella regione ammissibile del MP (i vincoli di questo problema sono costruiti a tale proposito[1]) e il valore della funzione obbiettivo del MP diminuisce, portando a soluzioni migliori rispetto a x^*.

Se invece questo problema è inammissibile, o se l'ottimo è non negativo, vorrà dire che nel semispazio $\{x_d > 0\}$ non c'è nulla di interesse. Possiamo considerare quindi l'altro semispazio $\{x_d < 0\}$ risolvendo il problema:

$$\begin{aligned} \min \quad & \sum_{j=1}^{d-1} c_j x_j \\ \text{soggetto a} \quad & \sum_{j=1}^{d-1} a_{ij} x_j - a_{id} \geq 0, \quad i \in I \end{aligned} \tag{2.7}$$

in cui al contrario si è posto $x_d = -1$. Analogamente a prima, il semispazio $\{x_d < 0\}$ contiene soluzioni migliori di x^* rispetto al MP se e solo se questo problema ha soluzioni il cui valore della funzione obbiettivo è negativo.

Si osserva che, essendo x^* ottimo sull'iperpiano, al più uno dei due problemi appena elencati può avere soluzione di valore negativo. Se nessuno dei due problemi ha una soluzione di questo tipo, x^* è ottimo globale per il MP e abbiamo concluso.

- CASO 2: il secondo caso avviene quando il problema è inammissibile sull'iperpiano $\{x_d = 0\}$ ed abbiamo dunque ottenuto ricorsivamente un vettore $x' = (x'_1, \ldots, x'_{d-1}, 0)$ che minimizza f definita prima su $\{x_d = 0\}$. La funzione è convessa, quindi se vogliamo vedere in quale semispazio si può trovare una soluzione ammissibile per il MP, basterà cercare in un intorno di x'. Definiamo

$$I' := \left\{ i \ : \ f(x'_1, \ldots, x'_{d-1}, 0) = b_i - \sum_{j=1}^{d-1} a_{ij} x'_j \right\}$$

che ricordando la definizione di f:

$$f(x_1, \ldots, x_d) = \max \left\{ b_i - \sum_{j=1}^{d} a_{ij} x_j \ : \ i = 1, \ldots, n \right\}$$

[1] Per i vincoli con $i \in I$, vale $\sum a_{ij} x_j^* = b_i$ soddisfatta ad uguaglianza, per cui una direzione che soddisfi $\sum_{j=1}^{d-1} a_{ij} x_j + a_{id} \geq 0$ porterà la nuova soluzione $(1-\varepsilon)x^* + \varepsilon x$ a soddisfare ancora i vincoli del MP.

rappresenta l'insieme degli indici che realizzano il massimo di f in x', che per condizione di inammissibilità dovrà essere > 0. Siamo interessati alle seguenti domande:

D1: esiste $y = (y_1, \ldots, y_d)$ tale che $y_d > 0$ e $\sum_{j=1}^{d} a_{ij} y_j > 0$ per gli $i \in I'$?

D2: esiste $z = (z_1, \ldots, z_d)$ tale che $z_d < 0$ e $\sum_{j=1}^{d} a_{ij} z_j > 0$ per gli $i \in I'$?

Una risposta affermativa a D1 implica che è possibile trovare una combinazione di x' e y che faccia diminuire il valore di f rispetto a quello che era in x'; tale combinazione dovrà avere coefficiente di y positivo perché questo accada, quindi la ricerca del minimo globale di f avverrà in $\{x_d > 0\}$. In modo analogo se è la risposta di D2 ad essere affermativa si procede guardando in $\{x_d < 0\}$. Se invece le risposte ad entrambe le domande sono negative, x' rimare il minimo globale per f, il che certifica l'inammissibilità del MP su tutto $\mathbb{R}^d$.

Risponderemo alle due domande con un metodo scaltro: consideriamo il problema $(d-1)$-dimensionale:

$$\begin{aligned} &\min \quad 0 \\ &\text{soggetto a} \quad \sum_{j=1}^{d-1} a_{ij} y_j \geq -a_{id}, \quad i \in I' \end{aligned} \tag{2.8}$$

Se questo è ammissibile, allora esiste un vettore y tale che $y_d > 0$ ($y_d = 1$) e $\sum_{j=1}^{d} a_{ij} y_j \geq 0$ ($i \in I'$). In tal caso la risposta a D2 è negativa: infatti se esistesse uno z con $z_d < 0$ e $\sum_{j=1}^{d} a_{ij} z_j > 0$ ($i \in I'$), allora esisterebbe anche un vettore x, combinazione convessa opportuna di y e z tale che $x_d = 0$ e per gli $i \in I'$ $\sum_{j=1}^{d} a_{ij} x_j > 0 \Rightarrow \sum_{j=1}^{d-1} a_{ij} x_j > 0$ contraddicendo la minimalità di x' per f su $\{x_d = 0\}$. Analogamente se esistesse uno z ammissibile per

$$\begin{aligned} &\min \quad 0 \\ &\text{soggetto a} \quad \sum_{j=1}^{d-1} a_{ij} z_j \geq a_{id}, \quad i \in I' \end{aligned} \tag{2.9}$$

la risposta a D1 sarebbe negativa.

Consideriamo dunque due insiemi di disuguaglianze in $d-1$ variabili:

$$\begin{aligned} &\sum_{j=1}^{d-1} a_{ij} y_j \geq -a_{id}, \quad i \in I' \quad (1) \\ &\sum_{j=1}^{d-1} a_{ij} z_j \geq a_{id}, \quad i \in I' \quad (2) \end{aligned} \tag{2.10}$$

– Se entrambi gli insiemi (1) e (2) sono ammissibili o inammissibili, entrambe le domande hanno esito negativo e quindi x' è il minimo globale per f che certifica l'inammissibilità del MP (se entrambi sono ammissibili, le risposte a D1 e D2 sono entrambe negative per quanto osservato

sopra, se entrambi sono inammissibili vuol dire che non esistono y/z tali che $\sum a_{ij}y_j \geq 0/\sum a_{ij}z_j \geq 0$, a maggior ragione non ne esisteranno che verifichino la disuguaglianza stretta).

– Se invece esattamente uno dei due insiemi è ammissibile, si può procedere con la ricerca nel semispazio corrispondente: se (1) è ammissibile solo la risposta a D1 è affermativa e si procede dunque in $\{x_d > 0\}$, se (2) è ammissibile si procede invece in $\{x_d < 0\}$.

Questo conclude la descrizione di come agisce l'oracolo.

L'oracolo può dover risolvere fino a tre problemi $(d-1)$-dimensionali di al più n vincoli infatti:

- si risolve il problema sull'iperpiano $\{x_d = 0\}$: di n vincoli e dimensione $(d-1)$;
- se tale problema è illimitato, è illimitato anche il MP e si conclude;
- se tale problema ammette ottimo (CASO 1) si possono dover risolvere fino a due PL $(d-1)$-dimensionali di vincoli in I;
- se tale problema è inammissibile (CASO 2) si possono dover risolvere fino a due PL $(d-1)$-dimensionali di vincoli in I'.

Va però notato che le cardinalità di I e I' sono al più d, e quindi almeno due PL su tre sono sempre molto semplici da risolvere.

2.6 Conclusione

Separiamo la trattazione dei due diversi approcci di ricerca multidimensionale riportati al termine della sezione 2.4.

- Se viene utilizzato l'approccio 1: risolvendo al più $3 \cdot 2^{d-1}$ PL di ordine $n \times (d-1)$ (ossia consultando $A(d) = 2^{d-1}$ volte l'oracolo), si può ridurre un problema di ordine $n \times d$ a uno di ordine $n(1-2^{1-2^d}) \times d$ (ossia passare da n vincoli a $(1-B)n$ come già visto). Quindi il costo totale di questo approccio, che denoteremo $LP_1(n,d)$ sarà:

  $$LP_1(n,d) \leq 3 \cdot 2^{d-1} LP_1(n,d-1) + LP_1(n(1-2^{1-2^d}),d) + \theta(nd)$$

  Dove la notazione utilizzata per θ e la sua dipendenza lineare in nd sono stati discussi nella parte finale della sezione 2.4. Proviamo per induzione doppia su n e d che per ogni d esiste una costante $C(d)$ tale che $LP_1(n,d) < C(d) \cdot n$ per ogni n; tale costante inoltre verifica:

  $$C(d) \leq 3 \cdot 2^{2^d+d-2} C(d-1)$$

  Dimostrazione. Nel caso base in cui $n = 1$ la condizione è banalmente verificata con una $C(d)$ anche migliore di quella dell'enunciato, poiché è sufficiente consultare una volta l'oracolo per avere l'informazione. Per $d = 1$ si ha, come visto in 2.4, $C(1) = 1$ che verifica la condizione.

Con un ragionamento analogo a quello fatto nella dimostrazione della sezione 2.4, si costruisce una griglia su $\mathbb{R}^2$, non più isomorfa ad $\mathbb{N}^2$ ma a un sottoreticolo di $\mathbb{Q}^2$ (costituito da $\mathbb{N}$ per quanto riguarda i d e da $\{\mathbb{N} \cup \mathbb{N}(1-2^{1-2^d}) \mid d \in \mathbb{N}\}$ per gli n). In questo sottoreticolo possiamo ancora una volta supporre di risolvere "per righe" dove si ha d in ascissa e n in ordinata: nel dimostrare (n,d) si può supporre vero il risultato per tutti gli (m,h) per $m \in \{\mathbb{N} \cup \mathbb{N}(1-2^{1-2^d}) \mid d \in \mathbb{N}\}, m < n$ e $\forall h \in \mathbb{N}$ e per gli (n,h) per $h \in \mathbb{N}, h < d$.

Sia quindi vera per ipotesi induttiva la tesi nei casi $(n,d-1)$ e $(n(1-2^{1-2^d}),d)$ e proviamola per (n,d).

$$LP_1(n,d) \leq 3 \cdot 2^{d-1} LP_1(n,d-1) + LP_1(n(1-2^{1-2^d}),d) + \theta(nd) \leq$$

$$\leq 3 \cdot 2^{d-1} C(d-1) \cdot n + C(d) \cdot n(1-2^{1-2^d}) + Knd =$$

$$= [3 \cdot 2^{d-1} C(d-1) + C(d) \cdot (1-2^{1-2^d}) + Kd]n =: C(d) \cdot n$$

dove $\exists K$ per definizione di θ. Si tratta ora di verificare che $C(d)$ rispetti la condizione espressa sopra:

$$3 \cdot 2^{d-1} C(d-1) + C(d) \cdot (1-2^{1-2^d}) + Kd = C(d) \Longleftrightarrow C(d) 2^{1-2^d} =$$

$$= 3 \cdot 2^{d-1} C(d-1) + Kd \Longleftrightarrow C(d) \leq 3 \frac{2^{d-1}}{2^{1-2^d}} C(d-1) = 3 \cdot 2^{2^d+d-2} C(d-1)$$

Dove il termine Kd viene semplicemente ignorato poiché ininfluente dato che l'altro addendo ha un 2^{d-1}. □

Si può osservare anche che $C(d) \leq 3 \cdot 2^{2^d+d-2} C(d-1)$ implica $C(d) < 2^{2^{d+2}}$ ovvero $C(d) = 2^{O(2^d)}$. Proviamo il risultato per induzione su d; ricordando cosa già ottenuto nella sezione 2.4, come caso base per l'approccio 1: $C(1) = 1$, si vede facilmente che esso verifica la condizione. Supposto vero per $d-1$:

$$C(d) \leq 3 \cdot 2^{2^d+d-2} C(d-1) < 3 \cdot 2^{2^d+d-2} 2^{2^{d-1+2}} =$$

$$= 3 \cdot 2^{2^d+d-2+2^{d+1}} < 2^{3 \cdot 2^d+d} < 2^{4 \cdot 2^d} = 2^{2^{d+2}}$$

- L'approccio 2 riduce il problema originale ad un problema sempre d-dimensionale ma in soli $\frac{n}{2}$ vincoli, interrogando rispetto la posizione degli $H_{ik}^{(1)}$ e degli $H_{ik}^{(2)}$. Ognuna di queste interrogazioni è nel caso peggiore costituita da 3 problemi $(d-1)$-dimensionali in n vincoli (anche se sappiamo che il numero di vincoli è molto minore per almeno 2 di questi PL su 3). Per quanto già visto a riguardo nella sezione 2.4, sono necessarie un numero di interrogazioni totali pari a

$$Q(n,d) \leq (\log_2 n)^d \frac{2^d}{(d-2)!} = O((\log_2 n)^d)$$

con un costo ulteriore pari a $T(n,d) = O(d2^d n)$ come mostrato al termine della sezione 2.4. Possiamo quindi dire che il costo totale ottenuto con l'approccio 2 è:

$$LP_2(n,d) \leq c\frac{(2\log n)^d}{(d-2)!}LP_2(n,d-1) + LP_2\left(\frac{n}{2},d\right) + O(d2^d n)$$

Proviamo che questo porta, per d fissato, a $LP_2(n,d) = O\left(n(\log n)^{d^2}\right)$

Dimostrazione. Ancora una volta proviamo la tesi con induzione doppia su n e d come già visto più volte. I casi base per $n = 1$ e $d = 1$ sono quelli visti in 2.4, sommando i rispettivi Q e T che rappresentano rispettivamente il numero di interrogazioni e il costo computazionale aggiuntivo nei rispettivi casi. Si ha dunque:

$$LP_2(n,1) = 1 + \lfloor \log_2 n \rfloor + n; \quad LP_2(1,d) = 1 + 1$$

Il 2 del caso $n = 1$ può creare qualche problema dato che $\log 1 = 0$, ma se al posto di $\log n$ mettessi un $\log(n+1)$ otterrei un costo asintoticamente uguale e che non crea problemi di questo tipo, tuttavia dato che si tratta di un caso solo e neanche particolarmente rilevante dato che molto elementare, posso mantenere il risultato enunciato senza dover fare correzioni. Per il caso $d = 1$ si ha

$$1 + \lfloor \log_2 n \rfloor + n \leq 1 + \frac{\log n}{\log 2} + n \leq C(1) n \log n$$

per n abbastanza grande, che prova che la tesi è verificata con $C(1) = 1$.

Supponiamo ora veri i casi $(n,d-1)$ e $\left(\frac{n}{2},d\right)$ e proviamo per (n,d):

$$LP_2(n,d) \leq c\frac{(2\log n)^d}{(d-2)!}C(d-1)\cdot\left(n(\log n)^{(d-1)^2}\right) + c\,C(d)\cdot\left(\frac{n}{2}\left(\log\frac{n}{2}\right)^{d^2}\right) + O(d2^d n)$$

Nelle ipotesi in cui mi trovo, cioè che d sia fissato, chiaramente $O(d2^d n)$ è anche un $O\left(n(\log n)^{d^2}\right)$ per cui posso ignorarlo. Un $\log\frac{n}{2}$ nel secondo addendo può essere maggiorato con il solo $\log n$. Infine la costante c ottenuta dallo scioglimento dell'O grande può essere omessa. Mi riduco quindi a:

$$LP_2(n,d) \leq \frac{(2\log n)^d}{(d-2)!}C(d-1)\left(n(\log n)^{(d-1)^2}\right) + C(d)\left(\frac{n}{2}(\log n)^{d^2}\right) =$$

$$= n(\log n)^{d+(d-1)^2}\cdot C(d-1)\frac{2^d}{(d-2)!} + n(\log n)^{d^2}\frac{C(d)}{2} <$$

$$< n(\log n)^{d^2}\cdot C(d-1)\frac{2^d}{(d-2)!} + n(\log n)^{d^2}\frac{C(d)}{2} =$$

$$= n(\log n)^{d^2}\cdot\left[C(d-1)\frac{2^d}{(d-2)!} + \frac{C(d)}{2}\right] =: n(\log n)^{d^2}\cdot C(d)$$

Ottenendo quindi costante $C(d)$ che soddisfa:

$$C(d) = C(d-1)\frac{2^d}{(d-2)!} + \frac{C(d)}{2} \iff \frac{C(d)}{2} = C(d-1)\frac{2^d}{(d-2)!} \iff$$

$$C(d) = \frac{2^{d+1}}{(d-2)!}C(d-1)$$

Che unito al fatto che $C(1) = 1$, ci dà:

$$C(d) = \prod_{i=1}^{d} \frac{2^{i+1}}{(i-2)!} = \frac{\prod_{i=1}^{d} 2^{i+1}}{\prod_{j=1}^{d-2} j!} = \frac{2^{\sum_{i=1}^{d} i+1}}{\prod_{j=1}^{d-2} j!} = \frac{2^{\frac{d^2+3d}{2}}}{\prod_{j=1}^{d-2} j!} < \frac{2^{d^2}}{\prod_{j=1}^{d-2} j!}$$

Si noti che nel primo passaggio della catena di uguaglianze, c'è un $(i-2)!$ definito a partire da $i = 1$ che potrebbe far sorgere il dubbio che si stia calcolando il fattoriale di -1; ciò però non sussiste, infatti una volta definito $C(1) = 1$ in partenza, si può dare la definizione per ricorsione a partire da 2, per cui il minimo $(i-2)!$ è per $i = 2$ che porta ad avere 0! che è ben definito. Si lascia la definizione iniziale con questa correzione solo a posteriori per lasciare l'idea intuitiva che ci sta alla base.

□

Per concludere si osservi che il secondo approccio è generalmente migliore del primo per n non troppo grandi: sebbene questo porti ad un tempo $LP_2(n,d) = O\left(n(\log n)^{d^2}\right)$ che è peggiore al crescere di n rispetto al $LP_1(n,d) < C(d) \cdot n$ del primo approccio, si ha che a fare la differenza sono le costanti: la seconda $C(d)$ è molto migliore di un 2^{d^2} poiché c'è un prodotto di fattoriali a denominatore, mentre la prima è superesponenziale dell'ordine di 2^{2^d}. Può convenire però utilizzare un metodo che sia un ibrido tra i due approcci, che riesca a valutare a seconda della situazione che si presenta, quale dei due sia conveniente.
Lo stesso Megiddo termina il suo articolo citando delle possibili modifiche che coinvolgano scelte aleatorie che possano migliorare il tempo di calcolo stimato, aprendo le porte ai prossimi due capitoli di questo testo, dedicati a Clarkson e a Seidel, che introdurranno proprio la componente randomica all'interno dei loro algoritmi.

Capitolo 3

Clarkson

3.1 Introduzione

Scopo di Clarkson è di trovare algoritmi per la risoluzione di Programmi Lineari (PL) che abbiano numero di vincoli maggiore del numero di variabili ($n > d$); come già visto l'interesse particolare è per algoritmi il cui costo computazionale cresca linearmente in n, anche se ciò comporta crescite molto rapide in termini di d. Come già Megiddo prima di lui [7], Clarkson è interessato a PL con numero di vincoli arbitrariamente grande ma dimensione, cioè d, relativamente piccola. Si parte dalla base dell'algoritmo di Megiddo, che, come visto nel capitolo 2, richiede $O(2^{2^d}n)$[1] operazioni nel caso pessimo, un costo lineare in termini di n e accettabile nella condizione che d sia piccolo, ma decisamente migliorabile nella dipendenza da d. L'idea di Megiddo, che sarà qui ripresa, è di scartare ricorsivamente insiemi relativamente grandi di vincoli che non sono determinanti nella localizzazione dell'ottimo. Successivi miglioramenti all'algoritmo di Megiddo hanno portato a ridurre il costo fino a $O(3^{d^2}n)$ [4], [5], ma è con l'intuizione di Dyer e Frieze [6] di introdurre una componente randomica che il costo dell'algoritmo subisce il primo netto miglioramento, diventando (in valore atteso) pari a $O(d^{3d}n)$.

Partendo dalla stessa base, sfruttando una scelta aleatoria all'interno dell'algoritmo, si arriverà in questo capitolo ad un costo atteso di $O(d^2n) + (\log n)O(d)^{\frac{d}{2}+O(1)} + O(d^4\sqrt{n}\log n)$ con le costanti che non dipendono da d. Si può notare che:

- il termine dominante nella dipendenza da n è $O(d^2n)$ che migliora decisamente rispetto ai precedenti bound dati;
- il secondo termine dipenderà dalla soluzione di $O(d^2\log n)$ PL con $O(d^2)$ vincoli e d variabili con il metodo del simplesso, ognuno dei quali risolto in tempo $O(d)^{\frac{d}{2}+O(1)}$;
- il terzo termine sarà discusso più avanti.

[1] il bound si riferisce al costo ottenuto con l'approccio 1 visto nella sezione conclusiva 2.6, che rappresenta come già detto la strategia migliore nel caso di n molto grandi.

3.2 Notazioni ed assunzioni

Il problema che considereremo avrà n vincoli e d variabili. Al generico vincolo (che identifica un semispazio di $\mathbb{R}^d$) ci riferiremo come ad H, l'insieme degli n vincoli sarà denotato $\mathscr{H}$. Il poliedro ottenuto dall'intersezione degli n semispazi in $\mathscr{H}$ sarà identificato come $\mathscr{P}_{\mathscr{H}}$.

In questo capitolo per una questione di semplicità di spiegazione considereremo un problema di massimizzazione della prima variabile, ossia x_1^*=max$\{x_1 \mid Ax \leq b\}$, del tutto equivalente a trovare max$\{c \cdot x \mid Ax \leq b\}$ dopo aver operato un opportuno cambio di coordinate. Può ovviamente capitare che il PL sia impossibile o illimitato, e in questi casi questa versione del problema può non avere soluzione; può anche capitare che il problema ammetta più di un elemento in $\mathscr{P}_{\mathscr{H}}$ con la coordinata x_1 massima.

- Innanzitutto per trovare una soluzione ammissibile del PL si procede con la fase 1: dopo aver operato cambi di segni nei vincoli fino ad esserci ricondotti ad avere $b \geq 0$, si risolve $max\{t \mid Ax + 1 \cdot t \leq b, t \leq 0\}$, del quale una soluzione ammissibile è presto data. Se l'ottimo di questo problema ausiliario è $t = 0$, il problema originale è ammissibile e una soluzione di base da cui partire ci è nota dalla risoluzione della fase 1; se $t < 0$ si ha che $\mathscr{P}_{\mathscr{H}} = \emptyset$ nel problema originale. D'ora in poi assumeremo sempre che il PL abbia una soluzione ammissibile $x_0 \in \mathscr{P}_{\mathscr{H}}$; in questo caso il problema $A(x - x_0) \leq b - Ax_0$ ha, operando un cambio di variabile $y = x - x_0$, $y = 0$ come punto ammissibile; inoltre un qualsiasi y ammissibile per questo problema dà un $y + x_0$ come punto ammissibile del PL originale. Quindi un algoritmo per la risoluzione di un PL che abbia $0 \in \mathscr{P}_{\mathscr{H}}$ può essere usato per risolvere un generico PL, inoltre questo non provoca aumento della complessità a livello asintotico.

  Assumeremo dunque $0 \in \mathscr{P}_{\mathscr{H}}$, cioè che $b \geq 0$.

- Se il PL è limitato, considereremo ottima la soluzione di ordine lessicografico maggiore tra quelle che lo risolvono, così da avere una scelta canonica che identifica univocamente l'ottimo.

- Se il PL è illimitato si può prendere come ottimo il raggio partente da 0 nella direzione $x^*(\hat{\mathscr{H}})$ dove $\hat{\mathscr{H}}$ è l'insieme dei vincoli $Ax \leq 0$ uniti a $x_1 = 1$. Ci è già noto che per dare un raggio che certifica l'illimitatezza di un PL è necessario avere una soluzione del sistema $Ax \leq 0$, $c^t x > 0$; questa seconda corrisponde (nella scelta di c che operiamo) a $x_1 > 0$ che può essere riscalata a $x_1 = 1$. Il PL così ottenuto è chiaramente limitato ed ha $0 \in \mathscr{P}_{\mathscr{H}}$ per cui ha soluzione unica, ossia un unico raggio (scelto canonicamente) risolvente il PL illimitato di partenza.

Abbiamo dunque una versione di un Programma Lineare che restituisce sempre un'unica soluzione ottima, sia essa un punto o un raggio, e che ha sempre $0 \in \mathscr{P}_{\mathscr{H}}$. La ricerca di tale ottimo può essere svolta da un algoritmo che esegue il metodo del simplesso, che verrà richiamato più volte nell'algoritmo di Clarkson quando si tratterà di risolvere sottoproblemi di dimensione abbastanza piccola.

3.3 L'algoritmo

In questa sezione saranno menzionati 4 diversi algoritmi:

- $x_s^*(\mathscr{H})$ un algoritmo che risolve i PL con il metodo del simplesso;
- $x_r^*(\mathscr{H})$ un algoritmo ricorsivo;
- $x_i^*(\mathscr{H})$ un algoritmo iterativo;
- $x_m^*(\mathscr{H})$ l'algoritmo finale, misto dei precedenti: è una versione del ricorsivo che usa l'iterativo per le chiamate ricorsive. Viene creato per dare un limite di tempo che risulterà $O(d^2 n)$ limitando il numero di chiamate a $x_s^*(\mathscr{H})$ presenti nel ricorsivo.

L'**ALGORITMO RICORSIVO** si basa sul fatto che l'ottimo è unico e può essere determinato da d vincoli, tra gli n presenti in $\mathscr{H}$, posti ad uguaglianza. Esiste cioè un sottoinsieme $\mathscr{H}^* \subset \mathscr{H}$ con $|\mathscr{H}^*| \leq d$ tale che l'ottimo di $\mathscr{H}^*$ sia lo stesso di $\mathscr{H}$, ossia $x^*(\mathscr{H}^*) = x^*(\mathscr{H})$. I rimanenti vincoli in $\mathscr{H}^* \setminus \mathscr{H}$ sono dunque ridondanti per quanto concerne trovare l'ottimo; l'idea ispirata da Megiddo è di scartare rapidamente i vincoli in eccesso, sfruttando però in questo caso la componente aleatoria.

L'algoritmo crea un sottoinsieme $V^* \subset \mathscr{H}$ in più fasi, in ognuna delle quali un insieme $V \subset \mathscr{H} \setminus V^*$ viene aggiunto a V^*. V ha due proprietà: $|V| \leq 2\sqrt{n}$ e V contiene un vincolo di $\mathscr{H}^*$ (Lemma 4.1). Dopo d iterazioni V^* conterrà $\mathscr{H}^*$ ed avrà $O(d\sqrt{n})$ elementi.

L'algoritmo poi elimina il sottoinsieme di vincoli ridondanti $\mathscr{H} \setminus V^*$ ($\subset \mathscr{H} \setminus \mathscr{H}^*$) e l'operazione viene ripetuta con V^* [2]. La ricorsione termina quando ho un "piccolo" insieme di vincoli su cui posso applicare l'algoritmo col metodo del simplesso $x_s^*(\mathscr{H})$.

PSEUDOCODICE

function : $x_r^*(\mathscr{H} :=$ insieme di semispazi$)$
return : $x^* :=$ ottimo del PL
$V^* \leftarrow \emptyset$; $\quad C_d \leftarrow 9d^2$
if $n \leq C_d$ **then return** : $x_s^*(\mathscr{H})$
repeat
  scegli $R \subset \mathscr{H} \setminus V^*$ a random, $|R| = r = d\sqrt{n}$;
  $x^* \leftarrow x_r^*(R \cup V^*)$;
  $V \leftarrow \{H \in \mathscr{H} \mid x^* \text{viola } H\}$
  if $|V| \leq 2\sqrt{n}$ **then** $V^* \leftarrow V^* \cup V$
until $V = \emptyset$
return : x^*
end function x^*

[2] Si legga dopo aver terminato la trattazione dell'algoritmo ricorsivo. Ripetere l'operazione con V^* non significa avere un'operazione che consiste nel gettare i vincoli ridondanti e ripetere il tutto da capo su V^* al posto di $\mathscr{H}$, bensì l'iterazione naturale eseguita dall'algoritmo quando $R \subset \mathscr{H} \setminus V^*$ contiene l'ultimo vincolo di $\mathscr{H}^*$ (o gli ultimi se ne contiene più di uno); la frase: "l'operazione viene ripetuta su V^*" corrisponde quindi alla ricerca ricorsiva nel passaggio "$x^* \leftarrow x_r^*(R \cup V^*)$". Il risultato sarà un x^* che non viola nessun vincolo, producendo quindi $V = \emptyset$ che porta l'algoritmo a terminare.

Si inizializzano $V^* = \emptyset$ e $C_d = 9d^2$ (vedremo il perché di questa scelta nella sezione 3.4). Se $n \leq C_d$ si calcola l'ottimo col metodo del simplesso. Se $n > C_d$, si ripete ciò che segue con $V^* = \emptyset$ all'inizio:

- si sceglie randomicamente $R \subset \mathscr{H} \setminus V^*$ di dimensione $r = d\sqrt{n}$, con ogni insieme di tale dimensione equiprobabile;
- x^* viene determinato ricorsivamente come ottimo sui vincoli di $R \cup V^*$: $x^* = x_r^*(R \cup V^*)$;
- V viene definito come l'insieme dei vincoli violati da x^*;

Se $|V| \leq 2\sqrt{n}$, V viene incluso in V^*, cioè V^* diventa $V^* \cup V$; vedremo nella sezione 3.4 che la scelta di $2\sqrt{n}$ avviene poiché

$$\sqrt{n} = d \cdot \frac{n}{r} \approx \mathbb{E}[|V|]$$

Se V è vuoto ritorno x^* come $x^*(\mathscr{H})$ poiché non viola nessuno dei vincoli di $\mathscr{H}$ ed è dunque l'ottimo del problema, sennò si genera un nuovo R e si ripete questa parte.

L'**ALGORITMO ITERATIVO** si basa su una tecnica di ripesamento iterativo: come nel ricorsivo un sottoinsieme R viene scelto in modo aleatorio e V è definito come l'insieme dei vincoli violati dall'ottimo di tale sottoinsieme; a differenza del precedente, a ogni vincolo $H \in \mathscr{H}$ viene assegnato un peso intero e nella scelta di R, ogni vincolo H ha probabilità di essere scelto proporzionale al suo peso. Dato che V contiene a ogni iterazione un vincolo di $\mathscr{H}^*$ (se V non vuoto), ha senso incrementare i pesi dei vincoli di V, ad esempio raddoppiandoli. Il processo viene iterato cosicché i vincoli di $\mathscr{H}^*$ avranno un peso relativo abbastanza grande da far sì che ad una certa iterazione l'insieme R che scegliamo randomicamente contenga $\mathscr{H}^*$ con probabilità quasi 1.

PSEUDOCODICE

function : $x_i^*(\mathscr{H} :=$ insieme di semispazi$)$
return : $x^* :=$ ottimo del PL
for $H \in \mathscr{H}$ **do** $w_H \leftarrow 1$ **od**; $C_d \leftarrow 9d^2$
if $n \leq C_d$ **then return** : $x_s^*(\mathscr{H})$
repeat
    scegli $R \subset \mathscr{H}$ a random, $|R| = r = C_d$;
    $x^* \leftarrow x_s^*(R)$;
    $V \leftarrow \{H \in \mathscr{H} \mid x^* \text{viola } H\}$
    if $w(V) \leq 2w(\mathscr{H})/(9d-1)$ **then**
        for $H \in V$ **do** $w_H \leftarrow 2w_H$;
    end if
until $V = \emptyset$;
return : x^*;
end function x^*

Ad ogni vincolo è assegnato inizialmente il peso 1; il peso di $H \in \mathscr{H}$ sarà indicato w_H, il peso di un sottoinsieme (ad esempio V) sarà $w(V) = \sum_{H \in V} w_H$. Il sottoinsieme R è scelto senza reinserimento dal multi-insieme corrispondente a $\mathscr{H}$ in cui ogni H compare w_H volte (si ricordi che in un multi-insieme è possibile avere ripetizioni di elementi). $H \in \mathscr{H}$ sarà dunque scelto con probabilità $\frac{w_H}{w(\mathscr{H})}$ corrispondente alla probabilità uniforme nel multi-insieme. (Se $\overline{H}$ viene scelto, il concetto di "senza reinserimento" si traduce nel poter scegliere di nuovo $\overline{H}$ con probabilità $\frac{w_{\overline{H}}-1}{w(\mathscr{H})}$ anziché $\frac{w_{\overline{H}}}{w(\mathscr{H})}$). Il sottoinsieme R risultante potrà quindi avere $|R| < r$ ma questo è un vantaggio, quindi potremo sempre assumere $|R| = r$ poiché analizziamo il caso peggiore.

Si noti che in $x_i^*(\mathscr{H})$, il caso base è lo stesso del ricorsivo, cambiano però la dimensione del sottoinsieme generato R e la condizione per cui V sia scelto perché i pesi dei suoi vincoli vengano raddoppiati, questa non è più una condizione sulla dimensione di V, ma sul suo peso (cioè la somma dei pesi dei vincoli che lo compongono); il perché di questa scelta sarà chiaro col Lemma 4.4.

3.4 Analisi del tempo di calcolo

Iniziamo ora a dimostrare che il valore atteso del tempo di calcolo dell'algoritmo $x_m^*(\mathscr{H})$ (sopra non analizzato perché consiste del corpo di $x_r^*(\mathscr{H})$ con la chiamata ricorsiva per trovare x^*: $x_r^*(R \cup V^*)$ sostituita da $x_i^*(R \cup V^*)$) è quello riportato all'inizio del capitolo.

La dimostrazione procederà per gradi: il Lemma 4.1 proverà che se $V \neq \emptyset$, allora V contiene un vincolo di $\mathscr{H}^*$; i Lemmi 4.2, 4.3 serviranno a provare che la probabilità che un'esecuzione del "repeat" sia un "successo", ovvero che V venga scelto per essere aggiunto a V^*/per raddoppiare i suoi pesi (a seconda che si lavori con $x_r^*(\mathscr{H})$ o $x_i^*(\mathscr{H})$) è maggiore di $\frac{1}{2}$; il Lemma 4.4 darà un valore atteso del costo di $x_i^*(\mathscr{H})$ e infine il Teorema 4.5 proverà il risultato.

LEMMA 4.1: nei due algoritmi, se $V \neq \emptyset$, allora V contiene un vincolo di $\mathscr{H}^*$.

(Nota che un vincolo di $\mathscr{H}^*$ contenuto anche in V non è sicuramente in V^* poiché questi sono soddisfatti da x^*, ottimo di $R \cup V^*$, per costruzione; quindi aggiungere V a V^* vorrà dire aggiungere almeno un nuovo vincolo di $\mathscr{H}^*$ per ogni iterazione riuscita)

Dimostrazione. Supponiamo che x^* sia un punto (il caso x^* raggio è analogo). Per assurdo $V \neq \emptyset$ e V non contenga vincoli di $\mathscr{H}^*$.

Introduciamo una relazione d'ordine tra punti di $\mathbb{R}^d$: $x \succeq y$ se $(x_1, \ldots, x_d) \geq (y_1, \ldots, y_d)$ ossia l'ordine lessicografico.

Poiché V non contiene vincoli di $\mathscr{H}^*$ e x^* dell'algoritmo rispetta tutti i vincoli tranne quelli di V, x^* soddisfa tutti i vincoli di $\mathscr{H}^*$ da cui $x^*(\mathscr{H}^*) \succeq x^*$, poiché la relazione $\succeq$ corrisponde, per come abbiamo preso c e la scelta canonica dell'ottimo, a chiedere tra due valori ammissibili x e y quale è il migliore per il PL dato.

Allo stesso modo vale l'implicazione contraria, poiché $R \cup V^* \subset \mathscr{H}$ e $x^* = x^*(R \cup V^*)$, allora vale $x^* \succeq x^*(\mathscr{H}) = x^*(\mathscr{H}^*)$.

Dato che la soluzione è unica si ha $x^* = x^*(\mathscr{H}) = x^*(\mathscr{H}^*)$, dunque x^* è ottimo di $\mathscr{H}$ per cui non viola nessun $H \in \mathscr{H}$ da cui $V = \emptyset$. $\lightning$

□

LEMMA 4.2: Dati $V^* \subset \mathscr{H}$ e $R \subset \mathscr{H} \setminus V^*$ sottoinsieme aleatorio di dimensione r; $|\mathscr{H} \setminus V^*| = n$. $V \subset \mathscr{H}$ sia l'insieme dei vincoli violati da $x^*(R \cup V^*)$; allora vale

$$\mathbb{E}(|V|) \leq \frac{d(n-r+1)}{r-d}$$

Si noti che n ha un'accezione diversa in questo punto; si preferisce comunque usare n invece che usare un'altra variabile perché chiaramente la quantità $|\mathscr{H} \setminus V^*|$ è maggiorata da n (inteso come numero di vincoli totali, ossia come per tutto il capitolo fino a questo momento) e poiché analizzo il caso peggiore posso compiere questa maggiorazione, e inoltre perché, come si vedrà nella dimostrazione, la scelta di n permetterà di riconoscere più chiaramente alcuni valori, specialmente riguardo alcuni metodi di conteggio.

Dimostrazione. L'idea intuitiva è che essendo $x^*(R \cup V^*)$ ottimo, in particolare ammissibile, esso non violerà nessun vincolo di $R \cup V^*$ e quindi violerà "pochi" vincoli di $\mathscr{H}$. L'intera dimostrazione coinvolge il caso dell'algoritmo ricorsivo, l'unico in cui sia rilevante il valore atteso di $|V|$. Un risultato analogo per l'iterativo sarà citato nella dimostrazione del lemma 4.3, e si dimostra analogamente a questo, solo con più attenzione dato che si dovrà sostituire i pesi dei vincoli alle semplici cardinalità di sottoinsiemi.

Assumeremo per ora che il problema sia **non degenere**, cioè che non esistano $d+1-k$ iperpiani (tra quelli definiti dagli n vincoli del problema) che si intersecano in uno spazio di dimensione k; in particolare $d+1$ iperpiani NON conterranno alcun punto in comune.

Definiamo due classi più ampie in cui vedere $x^*(R \cup V^*)$:

$$\mathscr{F}_{\mathscr{H}} = \{x^*(T \cup V^*) \mid T \subset \mathscr{H} \setminus V^*\} \qquad \mathscr{F}_R = \{x^*(T \cup V^*) \mid T \subset R\}$$

Ovviamente $\mathscr{F}_R \subset \mathscr{F}_{\mathscr{H}}$ e $x^*(R \cup V^*)$ è l'unico elemento (anche se venisse determinato da diversi sottoinsiemi di vincoli, il punto nello spazio corrispondente sarebbe uno solo) di $\mathscr{F}_R$ che soddisfa tutti i vincoli di R (è vero che la definizione è sui $T \subset R$ ma anche in R, come in $\mathscr{H}$ ci sono vincoli ridondanti, dato che $r > d$ in entrambi gli algoritmi).

Dato $x \in \mathscr{F}_{\mathscr{H}}$, sia $|x|$ il numero di vincoli di $\mathscr{H}$ violati da x e sia I_x la funzione indicatrice per x:

$$I_x = \begin{cases} 1 & \text{se} \quad x = x^* = x^*(R \cup V^*) \\ 0 & \text{altrimenti} \end{cases}$$

I_x è una variabile aleatoria che dipende dalla scelta randomica di R. Questa ci permette di calcolare esplicitamente il valore atteso della dimensione di V:

$$\mathbb{E}(|V|) = \mathbb{E}\left(\sum_{x \in \mathscr{F}_{\mathscr{H}}} |x| \cdot I_x\right) = \sum_{x \in \mathscr{F}_{\mathscr{H}}} |x| \cdot \mathbb{E}(I_x) = \sum_{x \in \mathscr{F}_{\mathscr{H}}} |x| \cdot \mathbb{P}_x$$

La prima uguaglianza vale poiché a destra ho una somma di $|x^*| \cdot 1 = |V|$ e tutti gli altri termini sono pari a 0, la seconda vale per proprietà del valore atteso, ricordando che $|x|$ è deterministico e l'ultima uguaglianza è banale se intendiamo per $\mathbb{P}_x$ la probabilità che $x = x^*$.

Si tratta ora di trovare $\mathbb{P}_x$; dato che ogni sottoinsieme di dimensione r ha la stessa probabilità di essere scelto, vi sono

$$\mathbb{P}_x \cdot \binom{n}{r} = \frac{\text{num di s.i. di dim } r \text{ che contengono } x^*}{\text{num di s.i. di dim } r} \cdot (\text{num di s.i. di dim } r)$$

sottoinsiemi R che contengono x^*. Si tratta ora di contare in un altro modo quanti sono tali sottoinsiemi R contenenti x^*.

Perché x sia x^*, esso deve appartenere ad $\mathscr{F}_R$ e deve soddisfare tutti i vincoli di R. Si consideri il minimo sottoinsieme di vincoli (rispetto alla cardinalità) $T \subset \mathscr{H} \setminus V^*$ (ricordando che x$\in \mathscr{F}_{\mathscr{H}}$) tale che $x = x^*(T \cup V^*)$. Perché $x \in \mathscr{F}_R$ allora deve essere $T \subset R$. Nell'ipotesi di non degenerazione un vertice del poliedro (x in questo caso) è definito da un unico set di vincoli e questi sono al più tanti quanti la dimensione dello spazio, dunque tale T è unico ed ha al più d vincoli al suo interno. Sia dunque detta $i_x = |T| \leq d$. La condizione $T \subset R$ è equivalente a chiedere che gli i_x vincoli in T appartengano ad R mentre i rimanenti $r - i_x$ vincoli di R siano tra i restanti $n - |x| - i_x$ vincoli rispettati da x in $\mathscr{H} \setminus V^*$ oltre a quelli di T (x rispetta $n - |x|$ vincoli, di questi i_x in T più ulteriori $n - |x| - i_x$). I sottoinsiemi R di cardinalità r che contengono x^* sono dunque tanti quanti i modi di scegliere gli $r - i_x$ vincoli che uniti agli i_x di T definiscono R tra gli $n - |x| - i_x$ rispettati da x e disponibili in $\mathscr{H} \setminus V^*$. Ovvero

$$\binom{n - |x| - i_x}{r - i_x}$$

che unito a quanto visto prima mi dà $\mathbb{P}_x$:

$$\mathbb{P}_x \cdot \binom{n}{r} = \binom{n - |x| - i_x}{r - i_x} \qquad \mathbb{P}_x = \frac{\binom{n-|x|-i_x}{r-i_x}}{\binom{n}{r}}$$

Ora:

$$\binom{n - |x| - i_x}{r - i_x} = \frac{(n - |x| - i_x)!}{(r - i_x)!(n - |x| - r)!} =$$

$$= \frac{(n - |x| - i_x)!}{(r - i_x - 1)!(n - |x| - r + 1)!} \cdot \frac{n - |x| - r + 1}{r - i_x} \leq \frac{n - r + 1}{r - d} \binom{n - |x| - i_x}{r - i_x - 1}$$

$$\Rightarrow \mathbb{E}(|V|) \leq \frac{n - r + 1}{r - d} \cdot \sum_{x \in \mathscr{F}_{\mathscr{H}}} |x| \cdot \frac{\binom{n-|x|-i_x}{r-i_x-1}}{\binom{n}{r}}$$

Dove la disuguaglianza nella seconda riga deriva dal fatto che $i_x \leq d \Rightarrow r - i_x \geq r - d$ e che $|x| \geq 0$.

La dimostrazione è terminata se si riesce a mostrare che la somma è al più d. Si noti che $\frac{\binom{n-|x|-i_x}{r-i_x-1}}{\binom{n}{r}}$ è la probabilità che x sia un elemento di $\mathscr{F}_R$ che viola esattamente

un vincolo di R: infatti quelli a numeratore sono i modi di prendere $r - i_x - 1$ tra gli r vincoli di R, che uniti ai i_x di T significa soddisfare esattamente $r-1$ tra gli r vincoli di R. La somma è dunque il valore atteso del numero delle $x \in \mathscr{F}_R$ che violano esattamente un vincolo di R. Tale numero è al più d poiché, detto H il vincolo che è violato, un $x \in \mathscr{F}_R$ che viola solo H tra i vincoli di R sarà $x^*(R \cup V^* \setminus \{H\})$ ma $x^*(R \cup V^* \setminus \{H\}) = x^*(R \cup V^*)$ tranne nei casi in cui H è uno dei d vincoli che determinano univocamente $x^*(R \cup V^*)$. Da cui la tesi nel caso non degenere.

Resta da provare il lemma nel caso in cui $\mathscr{H}$ sia **degenere**. Dato un PL degenere vogliamo mostrare che esiste una versione perturbata in cui $\mathbb{E}(|\widetilde{V}|) \geq \mathbb{E}(|V|)$ dove $|\widetilde{V}|$, $|V|$ sono le dimensioni dell'insieme dei vincoli violati da x^* nei problemi perturbato (non degenere) ed originale (degenere). L'idea è simile a quella usata per evitare ciclaggio nel metodo del simplesso. Nel problema perturbato il vettore b è sostituito da $b + (\varepsilon, \varepsilon^2, \ldots, \varepsilon^n)$ con $\varepsilon > 0$ piccolo ottenendo così un sistema non degenere nel senso espresso all'inizio della dimostrazione. (Una perturbazione simile viene aggiunta a $\hat{\mathscr{H}}$ nel caso illimitato).

Ad ogni $x \in \mathscr{F}_{\mathscr{H}}$ del problema originale è associato un sottoinsieme di $\mathscr{F}_{\mathscr{H}'}$ con $\mathscr{H}'$ insieme dei vincoli perturbati: perturbando un vertice degenere si vengono a creare diversi altri vertici; si pensi ad esempio al vertice di una piramide a k facce (cioè iperpiani); effettuando la perturbazione, ognuno di questi iperpiani si discosta di una potenza di ε rispetto a quanto era prima, per cui le intersezioni non sono più un unico vertice ma un insieme di diversi altri vertici. Dato $x \in \mathscr{F}_{\mathscr{H}}$ e $T \subset \mathscr{H}$ con $x = x^*(T \cup V^*)$ e detto T' il corrispettivo di T nel PL perturbato, l'ottimo $x^*(T' \cup V^*)$ appartiene a $\mathscr{F}_{\mathscr{H}'}$. Inoltre se $x = x^*(R \cup V^*)$, un x' associato ad x è l'ottimo del PL perturbato associato con ε piccolo. L'ottimo x' viola almeno tanti vincoli (in $\mathscr{H}'$) quanti erano violati da x (in $\mathscr{H}$). Questo perché per ε piccolo x viola anche $H + \varepsilon^i$ se viola H[3], quindi anche x' viola almeno altrettanti vincoli perturbati.

Essendo il numero di vincoli violati da x' maggiore, si ha che il valore atteso della dimensione di V nel sistema perturbato è maggiore o uguale di quella nel caso originale e degenere, e questa rispetta la maggiorazione richiesta per quanto visto nella prima parte di questa dimostrazione. Grazie a questo si conclude. □

Il lemma verrà usato per mostrare che ad ogni passo l'algoritmo progredisce positivamente. Formalizziamo un termine che abbiamo già usato varie volte: un'esecuzione del "repeat" sarà detta un successo o un'iterazione riuscita se $|V| \leq 2\sqrt{n}$ nel caso di $x_r^*(\mathscr{H})$ e analogamente in $x_i^*(\mathscr{H})$, $x_m^*(\mathscr{H})$ se è soddisfatta la condizione che fa crescere V^* o raddoppia i pesi di V a seconda dell'algoritmo in cui ci troviamo.

LEMMA 4.3: la probabilità che un'esecuzione del "repeat" sia un successo è almeno $\frac{1}{2}$, quindi sono necessarie in media al più due esecuzioni del loop per avere un successo.

[3] i vincoli di un PL sono disuguaglianze non strette, la violazione di una di queste da parte di una certa x significa che tale x soddisfa la disuguaglianza stretta complementare, ma allora se x soddisfa il complementare di H, soddisfa anche il complementare di $H + \varepsilon^i$ per ε abbastanza piccolo.

Dimostrazione. In $x_r^*(\mathscr{H})$ si ha $r \geq d\sqrt{n}$ e grazie al lemma precedente ho

$$\mathbb{E}(|V|) \leq \frac{d(n-r+1)}{r-d} \leq \sqrt{n}$$

dato che

$$d(n-r+1) \leq d(n-1-\sqrt{n}+1) = d(n-\sqrt{n}) \quad r-d \geq d\sqrt{n}-d = d(\sqrt{n}-1)$$

assumendo $r \geq 1+\sqrt{n}$, sempre verificato nell'unico caso rilevante per cui d e n sono almeno 2,

$$\Rightarrow \frac{d(n-r+1)}{r-d} \leq \frac{d(n-\sqrt{n})}{d(\sqrt{n}-1)} = \sqrt{n}$$

Applicando la disuguaglianza di Markov

$$\mathbb{P}(|V| > 2\mathbb{E}(|V|)) \leq \frac{\mathbb{E}(|V|)}{2\mathbb{E}(|V|)} = \frac{1}{2}$$

da cui

$$\mathbb{P}(|V| \leq 2\sqrt{n}) \geq \mathbb{P}(|V| \leq 2\mathbb{E}(|V|)) \geq 1-\frac{1}{2} = \frac{1}{2} \qquad \Rightarrow \quad \mathbb{P}(|V| \leq 2\sqrt{n}) \geq \frac{1}{2}$$

Nel caso di $x_i^*(\mathscr{H})$ si prende $V^* = \emptyset$ nel lemma 4.2 (V^* non compare in questo algoritmo) e si lavora con $\mathscr{H}$ e R multi-insiemi con $n = w(\mathscr{H})$ anziché $|\mathscr{H}|$; il lemma 4.2 continua a valere e così anche questo.

□

LEMMA 4.4: dato un PL con $b \geq 0$, l'algoritmo iterativo $x_i^*(\mathscr{H})$ richiede in media $O(d^2 n \log n) + (d \log n) O(d)^{\frac{d}{2}+O(1)}$ operazioni con le costanti che non dipendono da d.

Dimostrazione. Si mostra che il loop di $x_i^*(\mathscr{H})$ è eseguito in valore atteso $O(d \log n)$ volte, osservando che $w(\mathscr{H}^*)$ cresce molto più rapidamente di $w(\mathscr{H})$ e quindi dopo $O(d \log n)$ iterazioni riuscite ho o che $V = \emptyset$, e quindi il loop termina in $O(d \log n)$ iterazioni, o che $w(\mathscr{H}^*) > w(\mathscr{H})$ $\lightning$. Grazie al lemma 4.1, V contiene un vincolo di $\mathscr{H}^*$, quindi quando un'esecuzione del loop è un successo c'è un vincolo $H \in \mathscr{H}^*$ il cui peso raddoppia nella data esecuzione.

Detto $d' = |\mathscr{H}^*|$, dopo kd' esecuzioni riuscite del loop ho che $w(\mathscr{H}^*) = \sum_{H \in \mathscr{H}^*} w_H$ e $w_H = 2^{i_H}$ per qualche i_H e $\sum_{H \in \mathscr{H}^*} i_H \geq kd'$ poiché per ogni esecuzione riuscita c'è un vincolo che raddoppia ma può essere anche più di uno. Da cui $w(\mathscr{H}^*) \geq 2^k d'$, il minimo possibile è quando ad ogni iterazione raddoppia solo un vincolo e ognuno raddoppia esattamente k volte, se un vincolo raddoppiasse più volte degli altri evidentemente avrei un valore maggiore.

D'altra parte quando gli elementi di V raddoppiano, $w(\mathscr{H})$ aumenta di $w(V) \leq \frac{2w(\mathscr{H})}{9d-1}$. Cioè in un'esecuzione riuscita (le uniche in cui raddoppio i pesi) $w(\mathscr{H})' \leq (1+\frac{2}{9d-1})w(\mathscr{H})$.

Dopo kd' successi, ricordando che il peso complessivo iniziale di $w(\mathscr{H})$ è n dato che i pesi iniziali sono tutti 1:

$$w(\mathscr{H}) \leq \left(1+\frac{2}{9d-1}\right)^{kd'} n \leq e^{\frac{2kd'}{9d-1}} n$$

dove per la seconda disuguaglianza si è usato il fatto che $1+x \leq e^x \Rightarrow \quad (1+x)^m \leq e^{xm}$. Cerchiamo $\bar{k}$ tale che $\forall k \geq \bar{k}$ valga $w(\mathscr{H}^*) > w(\mathscr{H})$

$$w(\mathscr{H}^*) > w(\mathscr{H}) \quad \Leftrightarrow \quad 2^k d' > e^{\frac{2kd'}{9d-1}} n \quad \Leftrightarrow \quad e^{\log(2^k d')} > e^{\frac{2kd'}{9d-1}+\log n} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad k\cdot\log 2+\log d' > \frac{2kd'}{9d-1}+\log n \quad \Leftrightarrow \quad k\left(\log 2-\frac{2d'}{9d-1}\right) > \log n-\log d' = \log\frac{n}{d'}$$

$$\Leftrightarrow \quad k > \frac{\log\frac{n}{d'}}{\log 2-\frac{2d'}{9d-1}} =: \bar{k}$$

e quindi dopo $\bar{k}d' = O(d\cdot\log n)$ iterazioni riuscite V sarà $\emptyset$, e l'algoritmo sarà terminato. Verifichiamo questo risultato lavorando con con $k \leq \bar{k}$:

$$kd' \leq d'\cdot\log\frac{n}{d'}\cdot\left(\frac{1}{\log 2-\frac{2d'}{9d-1}}\right) = d'(\log n-\log d')\cdot\frac{9d-1}{(9d-1)\cdot\log 2-2d'} \leq$$

Ora poiché $|\mathscr{H}^*| = d' \leq d$

$$\leq d\cdot(\log n)\cdot\frac{9d-1}{(9d-1)\cdot\log 2-2d} - \frac{d\cdot\log d'\cdot(9d-1)}{(9d-1)\cdot\log 2-2d} <$$

è chiaro che nel secondo termine il numeratore è positivo poiché prodotto di termini positivi, il denominatore è uguale a $(9\log 2-2)d-\log 2 > 0$ per $d > 1$, quindi posso minorare il secondo termine con 0 e quindi maggiorare quanto scritto sopra con il solo primo termine

$$< d\cdot\log n\cdot\frac{1}{\log 2-\frac{2d}{9d-1}} < d\cdot\log n\cdot\frac{1}{\log 2-1} = O(d\cdot\log n)$$

dato che $\frac{2d}{9d-1} < 1$.

Resta da vedere il tempo necessario ad eseguire il loop. Vitter [12] dà un algoritmo per il campionamento aleatorio che si presta a generare insiemi come R in presenza di elementi dotati di peso in tempo $O(n)$, usando il fatto che $w(\mathscr{H}) = n^{O(1)}$ (dato che il peso dei vincoli è polinomiale in n).

V è generato in tempo $O(dn)$, dato che bisogna verificare n vincoli per x^* che ha d variabili.

L'algoritmo del simplesso impiega $d^{O(1)}$ operazioni per visitare un qualunque vertice di $\mathscr{P}_{\mathscr{H}}$ e visita tutti i vertici al più una volta.

Vogliamo vedere quanti sono i vertici del PL in questione, ricordando che ci interessa analizzare il caso peggiore, in cui si vengono a formare il numero maggiore di

vertici. Ogni vertice può essere determinato dall'intersezione di d iperpiani che delimitano un vincolo, o anche da un numero maggiore nel caso sia un vertice degenere; tuttavia poiché vogliamo andare a contare il numero massimo possibile di vertici che si vengono a creare, ci basta considerare il caso non degenere in cui per definire un vertice servono solo d iperpiani. Se ogni vertice è delimitato da d iperpiani, e ne ho al più C_d a disposizione (tanti quanto è la dimensione di R) il numero di vertici possibile sarà al più pari al numero di modi in cui è possibile scegliere un sottoinsieme di cardinalità d tra un insieme di C_d, pari a $\binom{C_d}{d}$.

Tuttavia l'autore dà un differente metodo di conteggio del numero dei vertici, che non si discosta troppo da quello segnalato e che porta ad avere come risultato $\binom{2C_d}{\lfloor \frac{d}{2} \rfloor}$ vertici totali; si noti come questo risultato sia migliore[4] di quanto ricavato in precedenza, questo perché non si era preso in considerazione il fatto che un singolo vincolo non può intervenire nella creazione di tutti i vertici per cui è stato contato, per una questione di posizione relativa tra i vari iperpiani.

Si ha dunque come bound per il simplesso pari a $\binom{2C_d}{\lfloor \frac{d}{2} \rfloor} \cdot d^{O(1)} \approx O(d)^{\frac{d}{2}+O(1)}$. Per dimostrare l'approssimazione procediamo ricordando le formule di Stirling:

$$\binom{n}{k} \leq n^k; \quad \log(n!) \approx n\log n - n$$

Si ha dunque (tralasciando il coefficiente 2·9 e la parte intera):

$$\binom{2C_d}{\lfloor \frac{d}{2} \rfloor} \approx \binom{d^2}{\frac{d}{2}} \leq \frac{d^{2\frac{d}{2}}}{(\frac{d}{2})!} = \frac{d^d}{(\frac{d}{2})!}$$

Calcoliamo il logaritmo di questa quantità, che è pari a

$$\log\left(\frac{d^d}{(\frac{d}{2})!}\right) = d\log d - \log\left(\frac{d}{2}\right)! \approx d\log d - \frac{d}{2}\log\left(\frac{d}{2}\right) + \frac{d}{2} =$$

$$= \log\left(\frac{d^d}{\frac{d}{2}^{\frac{d}{2}}}\right) + O(d) = \log\left(\frac{d^d}{d^{\frac{d}{2}}} \cdot 2^{\frac{d}{2}}\right) + O(d) = \log 2^{\frac{d}{2}} + \log d^{\frac{d}{2}} + O(d)$$

[4] Ripercorrendo gli stessi passaggi fatti partendo dal numero di vertici trovato col metodo appena citato, anziché da quello proposto dall'autore, si può osservare che con Stirling

$$\binom{C_d}{d} \approx \binom{d^2}{d} \leq \frac{d^{2d}}{d!}$$

successivamente, passando al logaritmo si ottiene

$$\log\left(\frac{d^{2d}}{d!}\right) = 2d\log d - \log d! \approx 2d\log d - d\log d + d = \log d^d + O(d)$$

da cui tornando alla quantità di partenza tramite esponenziale si ottiene $O(d)^{d+O(1)}$ anziché $O(d)^{\frac{d}{2}+O(1)}$, un valore decisamente peggiore rispetto a quello ottenuto a partire dal conteggio di Clarkson, ma accettabile ricordando che d è supposto piccolo, e che comunque questo coefficiente andrà a moltiplicare un fattore $\log n$, che asintoticamente sarà ininfluente rispetto al termine lineare in n.

Per tornare al valore di partenza devo applicare l'esponenziale che mi dà in totale l'approssimazione cercata ($\log 2^{\frac{d}{2}}$ e $O(d)$ danno contributo inferiore che potrà essere inserito nell'$O(d)^{O(1)}$)

Il loop quindi richiede $O(dn) + O(d)^{\frac{d}{2}+O(1)}$ operazioni.

Il tutto quindi chiede un tempo medio pari a $O(d \cdot \log n) \cdot [O(dn) + O(d)^{\frac{d}{2}+O(1)}]$ che corrisponde alla tesi.

□

TEOREMA 4.5: $x_m^*(\mathscr{H})$ richiede in media $O(d^2n) + (d^2 \log n) \cdot O(d)^{\frac{d}{2}+O(1)} + O(d^4\sqrt{n}\,\log n)$ operazioni.

Dimostrazione. Assumeremo $b > 0$. V^* cresce di al più $2\sqrt{n}$ ad ogni iterazione riuscita, e al più d di queste iterazioni sono necessarie. La dimensione di $R \cup V^*$ è al più $\sqrt{C_d n}$: $|V^*| \leq d \cdot 2\sqrt{n} \quad |R| = r = d\sqrt{n}$

$$|R \cup V^*| \leq d \cdot 2\sqrt{n} + d\sqrt{n} = 3 \cdot d\sqrt{n} = \sqrt{9d^2 n} = \sqrt{C_d n}$$

Si noti, che questa è la stima migliore che posso fare poiché R e V^* sono disgiunti.

Chiamo $T_m(n)$ il tempo medio in cui $x_m^*(\mathscr{H})$ risolve un PL con n vincoli e d variabili e analogamente $T_i(n)$ per $x_i^*(\mathscr{H})$. Ricordo che la probabilità che un'iterazione sia un successo è $\geq \frac{1}{2}$. Il tempo richiesto per trovare insiemi V abbastanza piccoli in un'esecuzione di $x_m^*(\mathscr{H})$ è al più $2T_i(\sqrt{C_d n})$ dove il 2 è giustificato dalla probabilità di avere un successo, mentre il resto è dovuto al fatto che $x_m^*(\mathscr{H})$ ha il corpo di $x_r^*(\mathscr{H})$ ma la chiamata fatta su $x_i^*(\mathscr{H})$, quindi $x_i^* = x_i^*(R \cup V^*)$ e il tempo per risolvere tale sistema è per definizione $T_i(|R \cup V^*|) \leq T_i(\sqrt{C_d n})$ per quanto visto sopra.

Il tempo richiesto a verificare che un dato vincolo sia soddisfatto dall'ottimo calcolato x_i^* è $O(d)$ (d moltiplicazioni per i coefficienti, $d-1$ somme e un confronto) per un totale di $O(d^2 n)$ tra tutti i vincoli (n) e tutte le iterazioni (d).

Per $n > C_d$ (cioè quando eseguo il loop) il valore atteso del tempo di calcolo è quindi al più $O(d^2n) + (d^2 \log n) \cdot O(d)^{\frac{d}{2}+O(1)} + O(d^4\sqrt{n}\,\log n)$, vediamo perché:

$$T_m(n) \leq 2d \cdot T_i(\sqrt{C_d n}) + O(d^2 n) =$$

$$= 2d \cdot [O(d^2\sqrt{9d^2 n}\,\log\sqrt{9d^2 n}) + (d\,\log\sqrt{9d^2 n})O(d)^{\frac{d}{2}+O(1)}] + O(d^2 n)$$

Dove ho usato il risultato del lemma 4.4 sul tempo medio di $x_i^*(\mathscr{H})$ applicato al numero di vincoli corrente: $\sqrt{C_d n}$. La dimostrazione è conclusa se dimostro:

$$I : 2d(d\,\log\sqrt{9d^2 n}) \leq d^2\,\log n$$

$$II : 2d \cdot O(d^2\sqrt{9d^2 n}\,\log\sqrt{9d^2 n}) \leq O(d^4\sqrt{n}\,\log n)$$

Vediamoli:

$$I :\; 2d(d\,\log\sqrt{9d^2 n}) = d \cdot d\,\log(9d^2 n) =$$

$$= d^2\,(\log(9d^2) + \log n) = 2d^2 \cdot \log(3d) + d^2 \cdot \log n =$$

$$= O(d^2 \log d) + O(d^2 \log n) = O(d^2 \log n) \leq \gamma(d^2 \log n)$$

dove sono stati usati la definizione di O grande per $\exists \gamma \in \mathbb{R}$ e $d < n$; il fatto di avere la presenza di γ costante non influisce sul risultato finale.

$$II: \ 2d \cdot O(d^2 \sqrt{9d^2 n} \ \log \sqrt{9d^2 n}) \leq d\gamma'(d^2 \cdot 3d\sqrt{n} \ \log(9d^2 n)) \leq$$

$$\leq d^4 \cdot 3\gamma' \sqrt{n} \log(9d^2 n) = O(d^4 \sqrt{n} \log n)$$

grazie alla definizione di O grande ($\exists \gamma' \in \mathbb{R}$), e nell'ultimo passaggio per ignorare il $9d^2$ sotto il logaritmo si procede come nel punto I.

□

La complessità computazionale enunciata nell'introduzione è quindi verificata.

Si noti come molte delle scelte sui parametri di confronto utilizzati negli algoritmi non erano state spiegate nelle sezioni di presentazione, e trovano una loro risposta solo all'interno delle dimostrazioni di questa sezione. Inoltre il funzionamento generale dell'algoritmo risulta chiaro appieno solo dopo averne analizzato il funzionamento all'interno di queste dimostrazioni. Ci troviamo di fronte quindi un algoritmo molto efficiente (sempre in via teorica secondo le richieste di questo testo), non particolarmente elaborato, ma che nasconde delle insidie che sono spiegate solo eseguendo un'attenta analisi di tutto ciò che succede all'interno.

Questa è la critica maggiore di Seidel, che presenterà un'idea per un algoritmo molto più semplice, che possa essere apprezzato senza la necessità di un'analisi approfondita come in questo caso, e che sarà trattato nel prossimo capitolo.

Capitolo 4

Seidel

4.1 Introduzione

Seidel propone un'idea per un algoritmo (non sarà presentato alcuno pseudocodice che concretizzi quanto spiegato nelle prossime pagine) che risolva Programmi Lineari (PL) di n vincoli in d variabili in tempo che cresca linearmente con n nel caso in cui la dimensione del problema, d, sia sufficientemente piccola. Viene ripresa l'idea di implementare una scelta randomica all'interno dell'algoritmo che, come nel caso di Clarkson (capitolo 3), velocizzerà notevolmente in valore atteso il tempo di calcolo rispetto ad algoritmi puramente deterministici costruiti per risolvere lo stesso tipo di problema come quello di Megiddo.

L'idea di Seidel è di presentare un algoritmo che sia semplice nell'idea tanto quanto nella spiegazione, evitando ciò che considera un difetto dell'algoritmo di Clarkson, il fatto che per essere capito appieno, sia necessario comprendere tutta l'analisi del tempo di calcolo che ci sta dietro, in cui sono annidate spiegazioni sul perché l'algoritmo adoperi certi valori di confronto come condizione necessaria a svolgere varie operazioni.

L'algoritmo presentato avrà un valore atteso di operazioni necessarie pari a $O(d!n)$; l'obbiettivo dell'autore di semplificare l'analisi del proprio algoritmo è pienamente riuscito, e questo ci permetterà di analizzare più nel dettaglio alcuni casi elementari che sono stati invece dati per scontati, per non appesantire troppo la trattazione, nei capitoli dedicati a Clarkson e Megiddo.

Ricordiamo la notazione utilizzata: il problema che considereremo avrà n vincoli e d variabili. Al generico vincolo (che identifica un semispazio di $\mathbb{R}^d$) ci riferiremo come ad H, l'insieme degli n vincoli sarà denotato $\mathscr{H}$. La funzione obbiettivo (che in questa sezione considereremo da massimizzare) è data da un vettore c. Il poliedro ottenuto dall'intersezione degli n semispazi in $\mathscr{H}$ sarà identificato come $\mathscr{P}_{\mathscr{H}}$.

4.2 L'algoritmo

Lo scopo dell'algoritmo sarà trovare un vettore ammissibile di $\mathbb{R}^d$: $v \in \mathscr{P}_{\mathscr{H}}$ che massimizza la funzione lineare descritta da c. Ciò è equivalente a chiedere che v appartenga all'iperpiano tangente a $\mathscr{P}_{\mathscr{H}}$ la cui perpendicolare uscente da $\mathscr{P}_{\mathscr{H}}$ è c. L'algoritmo si fonda su una scelta aleatoria così come fa quello di Clarkson.

Viene rimosso un vincolo H dall'insieme $\mathscr{H}$, scelto in maniera casuale, con ognuno degli n semispazi che ha la stessa probabilità di essere scelto. Viene risolto ricorsivamente il programma lineare così ottenuto, ancora in d variabili ma con soli $n-1$ vincoli $\mathscr{H}' := \mathscr{H} \setminus \{H\}$, trovandone l'ottimo che chiamiamo v'. I casi che si presentano ora sono due: se $v' \in H$, allora v' è ammissibile per il PL di partenza ed è l'ottimo di un PL con un vincolo in meno, dunque è l'ottimo del problema originale: $v = v'$. Se invece $v' \notin H$, l'ottimo appena trovato non è una soluzione ammissibile del problema di partenza e apparentemente calcolare ricorsivamente v' può sembrare una perdita di tempo. Ricordando però che l'ottimo di un PL è, come ogni soluzione di base, definito univocamente come intersezione di d degli n iperpiani che delimitano i semispazi, capiamo che, nel caso l'ottimo sui vincoli $\mathscr{P}_{\mathscr{H}'}$ non sia l'ottimo su tutto $\mathscr{P}_{\mathscr{H}}$, H scelto randomicamente all'inizio dell'algoritmo è proprio uno dei d vincoli il cui iperpiano di delimitazione identifica l'ottimo del problema. Detto h l'iperpiano che delimita il semispazio H abbiamo dunque $v \in h$. L'idea ora è di cercare l'ottimo v non più su tutto $\mathscr{P}_{\mathscr{H}} \in \mathbb{R}^d$ ma su un nuovo poliedro che andremo a definire solo su h.

Sia dunque $\overline{c}$ la proiezione ortogonale di c su h e $\overline{\mathscr{H}'} := \{G \cap h \,|\, G \in \mathscr{H}'\}$. v sarà dunque l'ottimo del problema in $d-1$ variabili (poiché lavoro su h iperpiano di $\mathbb{R}^d$) e $n-1$ vincoli (i G di $\mathscr{H}'$ "aggiustati" su h). Se il caso iniziale fosse stato 1-dimensionale, questo avrebbe potuto essere risolto in tempo $O(n)$, ma questo punto sarà approfondito in seguito.

Alcune domande sorgono spontanee, come gestire i casi di illimitatezza e inammissibilità? Vedremo che il caso in cui il problema è impossibile, cioè quando $\mathscr{P}_{\mathscr{H}} = \emptyset$, verrà trovato in automatico dall'algoritmo al suo termine, quando esso risolverà il caso 1-dimensionale il quale non produrrà soluzioni. Caso diverso per quanto riguarda l'illimitatezza: decidiamo che non siamo interessati alla soluzione su tutto $\mathbb{R}^d$ ma solo all'interno di un ipercubo B_α ($-\alpha \leq x \leq \alpha$). Questa strategia può sembrare sulle prime approssimativa, ma nella prossima sezione sarà presentata un'idea su come possa essere formalizzata una strategia che restituisca l'ottimo appartenente al Bounding Box solo nel caso in cui il PL sia limitato, mentre nel caso di illimitatezza lo stesso uso del Bounding Box B_α restituirà un raggio che la certifica.

L'uso del Bounding Box si rivelerà necessario anche per l'uscita dalla ricorsione: è chiaro che a ogni passaggio ricorsivo i parametri diminuiscono o restano invariati (nel calcolo di v' passo da d, n a d, $n-1$; nel caso in cui $v' \neq v$ il sistema su h ha parametri $d-1$, $n-1$), ed è quindi necessario chiedersi cosa fare quando ci si trova con $n = 0$ ossia senza vincoli o con $d = 1$, cioè nel caso 1-dimensionale. Nel primo caso l'ottimo sarà proprio un vertice di B_α, quello scelto concordemente ai segni delle coordinate di c, sempre tenendo a mente che vogliamo massimizzare la funzione: prenderò il vertice che ha $x_i = \alpha$ se $c_i \geq 0$ e $x_i = -\alpha$ sennò (la decisione di scegliere $x_i = \alpha$ anziché $x_i = -\alpha$ nel caso $c_i = 0$ è del tutto arbitraria, ma ciò deve essere specificato all'inizio

così da rendere canonica la scelta dell'ottimo). Poiché va verificato il segno di ognuna delle componenti di $c \in \mathbb{R}^d$, l'ottimo nel caso base è ottenuto in tempo $O(d)$. Il secondo caso prevede di intersecare tra loro gli n vincoli nel caso 1-dimensionale ottenendo in tempo $O(n)$ l'intervallo ammissibile del problema, per poi scegliere in base al segno di $c \in \mathbb{R}$ quale estremo è l'ottimo del problema (ammesso che l'intervallo sia non vuoto, nel qual caso si restituisce immediatamente l'inammissibilità del PL).

L'algoritmo di Seidel ha un valore atteso del tempo di calcolo dell'ottimo pari a $O(d!n)$, che sarà dimostrato a breve. Come nel caso dell'algoritmo di Clarkson parleremo di valore atteso del tempo di calcolo poiché la scelta iniziale di prendere un vincolo H da un insieme $\mathscr{H}$ è aleatoria, con ogni vincolo in tale insieme equiprobabile. Inoltre assumeremo che il PL di partenza e tutti i successivi sottoproblemi abbiano un'unica soluzione ottima, e che tale vertice ottimo sia l'intersezione di esattamente d iperpiani di delimitazione di semispazi in $\mathscr{H}$.

CLAIM: Sotto tale ipotesi il metodo proposto ha in media un tempo di esecuzione pari a $O(d!n)$

Dimostrazione. Per induzione su d.

Se $d = 1$, il problema può essere risolto in tempo $O(n)$ trovando l'intervallo ammissibile e scegliendo l'estremo in base al segno di c.

Se $d > 1$, basta mostrare che il valore atteso del tempo necessario a ottenere v da v' (calcolato ricorsivamente come sopra) è $O(d!)$. Se $v' = v$ ho già ottenuto v in $O(1)$, il caso interessante è quando $v' \neq v$. Questo caso si verifica con probabilità al più $\frac{d}{n}$, poiché questa è la probabilità con cui H scelto randomicamente all'inizio è uno dei d semispazi i cui iperpiani di delimitazione definiscono v, ma potrebbe essere che alcuni tra tali d iperpiani si trovino al di fuori di B_α. Per ipotesi induttiva, il tempo atteso a risolvere il problema nel "caso sfortunato" $(d-1)$-dimensionale con $n-1$ vincoli è $O((d-1)!(n-1))$ da cui il tempo atteso a ottenere v' da v è $\frac{d}{n}\, O((d-1)!(n-1)) = O(d!)$. Ottenuto v da v' in $O(d!)$ operazioni, mi sono ricondotto a un problema di parametri $(n-1,d)$ da uno con (n,d). Iterando posso ottenere v' da un v'' ottimo del sistema in cui tolgo un ulteriore vincolo ottenendo dunque parametri $(n-2,d)$ con un ulteriore costo $O(d!)$; si può ripetere fino a raggiungere un unico vincolo, (caso $n = 0$) che può essere risolto in tempo $O(d)$. Poiché a ogni regressione del numero di vincoli ho un costo di $O(d!)$ e ho n di queste regressioni, il costo totale sarà pari a $nO(d!)O(d) = O(nd!)$. □

NOTA SULLE ASSUNZIONI FATTE:

- La richiesta che la soluzione ottima v sia unica è risolta canonicamente scegliendo come ottimo tra i vertici di $B_\alpha \cap \mathscr{P}_\mathscr{H}$ che massimizzano il prodotto con c quello che ha la rappresentazione lessicografica maggiore ordinata rispetto $(x_1, x_2, \ldots, x_d)$. Si tratta cioè di scegliere tra i vertici ottimi quello di coordinata x_1 maggiore, a parità di questa quello con la coordinata x_2 maggiore e così via fino alla d-sima coordinata che dovrà essere diversa o avrei lo stesso punto. Nulla vieta ovviamente di fare un'altra scelta sui criteri con i quali scegliere l'ottimo se ciò è necessario, purché la scelta sia canonica rispetto ad $\mathscr{H}$ come nell'esempio

appena fatto. Si noti infatti che per l'analisi del tempo di calcolo dell'algoritmo è necessario che v sia definito univocamente e in maniera canonica.

- L'assunzione che v sia l'intersezione di esattamente d iperpiani di delimitazione può essere scartata senza problemi. Resta il fatto che tra tutti i semispazi di $\mathscr{H}$ al più d hanno la proprietà che l'ottimo di $\mathscr{H} \setminus \{H\}$ sia diverso da v, e poiché l'analisi del tempo di calcolo di un algoritmo è un caso peggiore, ciò non influisce su quanto appena dimostrato.

Formalizziamo col prossimo risultato quanto detto finora.

Teorema: Col procedimento aleatorio sopra descritto un PL con n vincoli in d variabili può essere risolto in media in tempo $O(d!n)$.

Dimostrazione. Analizziamo come prima cosa i tempi di calcolo di alcune operazioni che saranno necessarie:

- verificare che un punto appartenga ad un semispazio è risolto in tempo $O(d)$: verificare corrisponde a fare d moltiplicazioni (ogni variabile per il suo coefficiente nel vincolo), $d-1$ somme tra tutti i termini, e il confronto $(\leq, =, \geq)$;

- proiettare ortogonalmente un d-vettore su un iperpiano impiega $O(d)$ operazioni: posso trovare n la normale all'iperpiano in tempo $O(1)$ nota l'equazione di questo; proiettare v su h significa trovare la combinazione di v con la normale n che appartenga ad h, ossia determinare λ: $(v+\lambda n) \in h$ equivalente a $n \cdot (v+\lambda n) = 0$ ossia
$$n \cdot v + \lambda \|n\|^2 = 0 \quad \Rightarrow \quad \lambda = -\frac{n \cdot v}{\|n\|^2}$$
Il prodotto scalare è operato in tempo $O(d)$ così come $\|n\|^2$ in quanto d prodotti e una somma, la divisione e la moltiplicazione per -1 sono operazioni singole. La proiezione sarà $v+\lambda n$ ottenuta con altre d moltiplicazioni e d addizioni. In tutto $O(d)$ operazioni;

- determinare l'intersezione tra un semispazio in $\mathbb{R}^d$ e un iperpiano impiega tempo $O(d)$: si sostituisce l'equazione dell'iperpiano della disequazione del semispazio ottenendo una disequazione con $d-1$ variabili. Come prima cosa isolare la variabile da eliminare è fatto in $O(d)$ operazioni, trasportando il termine dall'altra parte dell'uguale e poi effettuando al più d divisioni per il suo coefficiente. Per la sostituzione dovrò moltiplicare il coefficiente della variabile da eliminare per al più i $d-1$ termini corrispondenti alle restanti variabili e poi fare $d-1$ somme, per un totale complessivo di $O(d)$ operazioni.

Il valore atteso del tempo di calcolo dell'algoritmo $T(d,n)$ soddisfa quindi:
$$T(d,n) \leq \begin{cases} O(n) & \text{se } d = 1, \\ O(d) & \text{se } n = 0, \\ T(d,n-1) + O(d) + \frac{d}{n}O(dn) + \frac{d}{n}T(d-1,n-1) & \text{altrimenti} \end{cases}$$
Questo perché:

- se $d = 1$ il tempo richiesto è già stato dimostrato essere pari a $O(n)$;
- se $n = 0$ basta trovare il vertice del Bounding Box che massimizza il prodotto con c, in tempo $O(d)$ come già dimostrato;
- nel caso generale è necessario togliere l'iperpiano H con costo 1 (è possibile dare algoritmi che scelgono un elemento da un insieme di cardinalità n in tempo che non influisce su quanto stiamo calcolando); si trova ricorsivamente v' ottimo di $\mathscr{P}_{\mathscr{H}'}$ in tempo $T(d,n-1)$ per come questo valore è definito; successivamente si verifica se $v' \in h$ in tempo $O(d)$ come detto sopra. Se $v' = v$ ho finito, se ciò non accade (con probabilità $\frac{d}{n}$) si costruisce $\overline{\mathscr{H}'}$:=$\{G \cap \text{h} \,|\text{G} \in \mathscr{H}'\}$ intersecando $n-1$ volte un semispazio (G) con l'iperpiano h, ciò richiede $(n-1)O(d) = O(dn)$ operazioni, si proietta c sull'iperpiano h che delimita H ottenendo $\overline{c}$ in tempo $O(d)$ come già visto sopra (questo ulteriore $\frac{d}{n}O(d)$ può essere visto come già compreso nel $\frac{d}{n}O(dn)$ dell'intersezione tra semispazi e iperpiani, per questo non compare). Infine si risolve il sistema di vincoli $\overline{\mathscr{H}'}$ e funzione obiettivo $\overline{c}$ in tempo $T(d-1,n-1)$ da cui la validità del sistema.

Da questo si deduce

$$T(d,n) = O\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!n\right)$$

Vediamolo per induzione (doppia su n, d, come già svolta in molte dimostrazioni analoghe nel capitolo 2, i particolare sezioni 2.4, 2.6): i casi base sono banalmente verificati. Valgano

$$T(d,n-1) \leq C\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!(n-1)\right); \quad T(d-1,n-1) \leq C\left(\sum_{i=1}^{d-1}\left(\frac{i^2}{i!}\right)(d-1)!(n-1)\right)$$

$\exists C$ per definizione di O grande; moltiplicata la seconda per $\frac{d}{n}$ come nel sistema ho

$$\frac{d}{n}T(d-1,n-1) \leq \frac{d}{n}C\left(\sum_{i=1}^{d-1}\left(\frac{i^2}{i!}\right)(d-1)!(n-1)\right) = C\left(\sum_{i=1}^{d-1}\left(\frac{i^2}{i!}\right)d!\frac{n-1}{n}\right) \leq$$

$$\leq C\left(\sum_{i=1}^{d-1}\left(\frac{i^2}{i!}\right)d!\right) < C\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!\right)$$

Sommando ora $T(d,n-1) + \frac{d}{n}T(d-1,n-1)$ ottengo

$$T(d,n-1) + \frac{d}{n}T(d-1,n-1) < C\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!(n-1+1)\right) = O\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!n\right)$$

Inoltre $O(d) \leq O(\sum_{i=1}^{d}(\frac{i^2}{i!})d!n)$ banalmente e $\frac{d}{n}\,O(dn) = \gamma d^2 \leq \gamma\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!n\right) \quad \exists\gamma$ poiché:

$$\gamma\left(\sum_{i=1}^{d}\left(\frac{i^2}{i!}\right)d!n\right) \geq \gamma\left(\frac{d^2}{d!}d!n\right) \geq \gamma d^2$$

Ovviamente $O(n)$ ed $O(d)$ sono anche $O(\sum_{i=1}^{d}(\frac{i^2}{i!})\, d!n)$

Infine, poiché $\frac{i^2}{i!}$ è definitivamente minore di $\frac{1}{i^2}$, la cui serie converge, anche la somma converge per $d \to \infty$ a un k finito. Quindi $O(\sum_{i=1}^{d}(\frac{i^2}{i!})\, d!n) = O(d!n)$. □

4.3 L'uso del Bounding Box

Sono dovute al lettore delle spiegazioni riguardo all'uso del Bounding Box B_α che forza la limitatezza del problema imponendo lower/upper bound alle variabili; si potrebbe obbiettare che:

- α potrebbe essere scelto troppo piccolo cosicché B_α non conterrebbe l'ottimo di $\mathscr{P}_\mathscr{H}$;
- la scelta di α potrebbe essere importante nel determinare se $\mathscr{P}_\mathscr{H}$ è illimitato nella direzione della funzione obbiettivo c.

Proponiamo due modi di risolvere questi problemi:

1. Senza introdurre il Bounding Box, aggiustiamo la definizione di ottimo del PL: se $\mathscr{P}_\mathscr{H}$ è limitato nella direzione di c, scegliamo come ottimo il vertice di $\mathscr{P}_\mathscr{H}$ con l'ordine lessicografico maggiore (o un'altra scelta che sia canonica) tra quelli che massimizzano il prodotto scalare con c; sennò, in caso di non limitatezza, occorre operare una scelta canonica tra i raggi appartenenti al cono di recessione di $\mathscr{P}_\mathscr{H}$ nel quale il vettore unità massimizza il prodotto scalare con c. (Per quanto riguarda la scelta canonica in questo caso si ricordi come Clarkson risolve lo stesso problema di unicità dell'ottimo, anche nel caso di illimitatezza: aggiungendo il vincolo $c \cdot v = 1$, che per la scelta di c nel Clarkson abbiamo visto scritta $x_1 = 1$, e trattando questo nuovo problema limitato con la scelta canonica citata nel caso analogo, sezione 3.2).

2. Continuiamo ad usare il Bounding Box B_α, non scegliendo esplicitamente α, ma usando un certo parametro λ maggiore di qualsiasi numero che compare nel problema. In questo modo le coordinate dei risultati che ho nei passaggi intermedi e finali del procedimento risulteranno polinomiali in λ di grado 1, poiché anche i vincoli saranno riscritti in forma parametrica. In particolare lo stesso ottimo sarà del tipo $v(\lambda) = u + \lambda w$ con u e w d-vettori. Per tutti i valori abbastanza grandi di λ, il vettore $v(\lambda)$ è l'ottimo di $B_\lambda \cap \mathscr{P}_\mathscr{H}$. Quindi se w è il vettore nullo, il problema è limitato e u è il vertice ottimo di $\mathscr{P}_\mathscr{H}$; sennò esiste un numero reale λ_0 tale che $\{v(\lambda) \mid \lambda \geq \lambda_0\}$ è un raggio contenuto in $\mathscr{P}_\mathscr{H}$ che certifica l'illimitatezza del problema.

L'implementazione che segue, che va a giustificare quanto appena detto, è interessante solo da un punto di vista teorico poiché mostra come l'uso del Bounding Box dia sempre il risultato esatto di un PL; Seidel riporta un algoritmo che chiameremo funzione LP che genera la coppia di vettori u, w la cui descrizione esula dagli intenti di questo testo. L'obbiettivo delle prossime righe sarà di descrivere qualitativamente il dato algoritmo, che ha solo una valenza teorica: infatti esso è più lento di quello descritto nel corpo di

questo capitolo, per cui ignoreremo direttamente il calcolo della sua complessità e ci limiteremo ad apprezzarne la costruzione.

I dati di Input sono $d > 0$ la dimensione del problema necessariamente intera, un d-vettore c=($c_1, \ldots, c_d$), un insieme A di $(d+2)$-vettori. L'Output può essere una coppia di d-vettori u, w con la proprietà che, per reali abbastanza grandi λ, il d-vettore $u + \lambda w$ sia il maggiore (in senso lessicografico) vettore x che massimizza $\sum_{i=1}^{d} c_i x_i$ e che rispetta i vincoli parametrizzati

$$\begin{aligned} \sum_{i=1}^{d} a_i x_i \leq a_{d+1} + \lambda a_{d+2} \quad & \forall a = (a_1, \ldots, a_{d+2}) \in A \\ -\lambda \leq x_i \leq \lambda \quad & \forall i \in \{1, \ldots, d\} \end{aligned} \tag{4.1}$$

Oppure l'Output può essere un certificato di inammissibilità del PL se $\forall \lambda \geq \lambda_0$ con $\lambda_0 \in \mathbb{R}$ i vincoli parametrizzati non hanno soluzione comune.

Si noti che i vincoli $-\lambda \leq x_i \leq \lambda$ impongono che il PL sia inammissibile o che ammetta soluzione ottima, non è possibile che il problema sia illimitato.

Noi siamo interessati alla soluzione di un PL del tipo

$$\begin{aligned} \max \quad & \sum_{i=1}^{d} c_i x_i \\ \text{soggetto a} \quad & \sum_{i=1}^{d} a_i x_i \leq a_{d+1}, \quad \forall a = (a_1, \ldots, a_{d+1}) \in A \end{aligned} \tag{4.2}$$

con A insieme di $(d+1)$-vettori. (Questo PL può essere illimitato)

Tale programma può essere risolto dalla funzione LP, se estendiamo ogni $a \in A$ a un $(d+2)$-vettore con la $(d+2)$-esima coordinata 0. Formalmente ora i vincoli parametrizzati sono

$$\sum_{i=1}^{d} a_i x_i \leq a_{d+1} + \lambda \cdot 0$$

Se fornisco un tale A, assieme alla dimensione d e alla funzione obbiettivo c, $LP(d, c, A)$ può restituire:

- inammissibilità, e allora anche il PL originale è inammissibile;
- una coppia di d-vettori (u, w) con $w = 0$, allora il PL originale è limitato e u è l'ottimo;
- una coppia di d-vettori (u, w) con $w \neq 0$, allora il PL originale è illimitato ed esiste λ_0 tale che il raggio $\{u + \lambda w) \mid \lambda \geq \lambda_0\}$ è tutto contenuto nella regione ammissibile $\mathscr{P}_{\mathscr{H}}$.

Capitolo 5

Conclusione

I tre metodi mostrati sono tra loro molto diversi, sia nel carattere espositivo che nel risultato raggiunto da ognuno. Questo mostra come dall'idea di voler conoscere i vincoli che determinano in maniera univoca l'ottimo di un PL, possono scaturire tecniche completamente differenti su come questa possa essere raggiunta.

Al di là dell'interesse accademico che mi ha coinvolto nella lettura dei tre articoli e nella stesura di questo testo, volto principalmente alla ricerca di un algoritmo efficiente in una vasta gamma di problemi, credo che alcune delle idee trattate, o anche solo accennate dagli autori possano essere d'aiuto nella risoluzione di altri problemi aperti o che devono ancora essere formulati, per cui meritano di essere ricordati qui così da poter essere approfonditi dal lettore:

- Algoritmi per la ricerca della mediana in tempo lineare nel numero di elementi, citato da Megiddo in 2.3, [1], [10];
- la costruzione di un oracolo che sappia dire la posizione della soluzione di un PL, nei tre casi in cui questa si può manifestare, rispetto un iperpiano, come ampiamente trattato in 2.5;
- l'idea di aggiungere una perturbazione in caso di sistema degenere come usato nella dimostrazione del Lemma 4.2 della sezione 3.4, che trasforma un vertice della regione degenere in un'insieme di vertici dati dall'intersezione dei vincoli perturbati;
- l'intuizione di usare il Bounding Box così come spiegato in 4.3, il che permette di escludere il caso dell'illimitatezza e di dare semplici condizioni per i casi base; al contempo questo può essere generalizzato così da poter fornire correttamente l'ottimo o il raggio per l'illimitatezza del problema, a seconda che il PL ammetta ottimo o sia illimitato.

Bibliografia

[1] A. V. Aho, J.E. Hopcroft e J.D. Ullman. *The Design and Analysis of Computer Algorithms*. A cura di Prentice Hall. 1974.

[2] J. L. Bentley. "Multidimensional divide-and-conquer". In: *Commun ACM* 23 (4 apr. 1980), pp. 214–229. DOI: 10.1145/358841.358850. URL: https://dl.acm.org/doi/abs/10.1145/358841.358850.

[3] K. Clarkson. "Las Vegas Algorithms for Linear and Integer Programming When the Dimension is Small". In: *Journal of the Association for Computing Machinery* 31 (1 ott. 1989), pp. 114–127. DOI: 10.1145/201019.201036. URL: https://dl.acm.org/doi/abs/10.1145/201019.201036.

[4] K.L. Clarkson. "Linear programming in $O(n3^{d^2})$ time". In: *Information Processing Letters* 15 (1986), pp. 725–738. DOI: 10.1137/0215052. URL: https://epubs.siam.org/doi/10.1137/0215052.

[5] M. E. Dyer. "On a multidimensional search technique and its application to the euclidean one-centre problem". In: *SIAM Journal on Computing* 44 (1989), pp. 203–212. DOI: 10.1137/0215052. URL: https://epubs.siam.org/doi/abs/10.1137/0215052.

[6] M. E. Dyer e A. M. Frieze. "A randomized algorithm for fixed-dimensional linear programming". In: *Mathematical Programming* 44 (1989), pp. 203–212. DOI: 10.1007/BF01587088. URL: https://link.springer.com/article/10.1007/BF01587088.

[7] N. Megiddo. "Linear Programming in Linear Time When the Dimension Is Fixed". In: *Journal of the Association for Computing Machinery* 31 (1 gen. 1984), pp. 114–127. DOI: 10.1145/2422.322418. URL: https://dl.acm.org/doi/pdf/10.1145/2422.322418.

[8] N. Megiddo. "Towards a genuinely polynomial algorithm for linear programming". In: *SIAM Journal on Computing* 12 (2 mag. 1983), pp. 347–353. DOI: 10.1137/0212022. URL: https://epubs.siam.org/doi/abs/10.1137/0212022.

[9] L. Monier. "Combinatorial solutions of multidimensional divide-and-conquer recurrences". In: *J. Algorithms* 1 (1980), pp. 60–74. URL: https://www.sciencedirect.com/science/article/abs/pii/019667748090005X.

[10] A. Schonage, M. Paterson e Pippenger N. “Finding the median”. In: *J. Comput. Syst. Sci.* 13 (1976), pp. 184–199. URL: http://wrap.warwick.ac.uk/59399/1/WRAP_Paterson_cs-rr-006.pdf.

[11] R. Seidel. “Small-Dimensional Linear Programming and Convex Hulls Made Easy”. In: *Discrete & Computational Geometry* 6 (1 1991), pp. 423–433. DOI: 10.1007/BF02574699. URL: https://link.springer.com/article/10.1007/BF02574699.

[12] J. S. Vitter. “Faster methods for random sampling”. In: *Communications of the ACM* (1984), pp. 703–718. DOI: 10.1145/358105.893. URL: https://dl.acm.org/doi/abs/10.1145/358105.893.