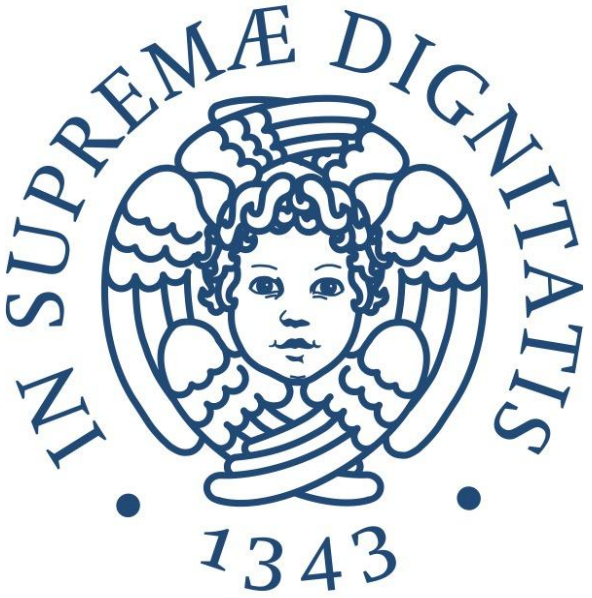


UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Risoluzione di problemi di soddisfacibilità in presenza di domini descritti da disuguaglianze lineari strette

Relatore: Candidato:

Prof. Marco Cococcioni **Gabriele Shu**

a.a. 2022-2023

Indice

Abstract

In questa tesi triennale è stata affrontata la risolubilità di un problema definito da un sistema di disequazioni lineari strette del tipo $\mathbf{A_s\ x < b_s}$. Si tratta dunque di un problema di soddisfacibilità, che a differenza dei ben noti problemi di soddisfacibilità booleana (SAT) o problemi di soddisfacibilità descritti da sistemi di disequazioni lineari non strette (Linear Programming Problems), i quali utilizzano tecniche di risoluzione già affermate (come l'algoritmo del simplesso o l'algoritmo del punto interno), non ha una specifica tecnica di risoluzione. Il problema in questione però è riconducibile ad una classe di problemi definiti problemi Lessicografici Multi-Obiettivo di Linear Programming (LMOLP), i quali grazie a recenti ricerche di matematica applicata e computazionale, possiedono due diversi approcci di risoluzione. L'approccio più intuitivo, detto anche approccio preemptive, consiste nel considerare il problema multi-obiettivo lessicografico come una serie di problemi a singolo-obiettivo risolti in sequenza. Il secondo approccio, basato su un concetto matematico, il grossone (un numero maggiore di qualsiasi altro numero finito), ha reso possibile l'implementazione di un algoritmo basato sul simplesso (Gross-Simplex) che permette di risolvere problemi dove la funzione obiettivo è costituita sia da parti finite che da numeri infinitesimali. L'obiettivo di questa tesi è di dimostrare che è possibile risolvere il problema utilizzando il primo approccio, eseguendo iterativamente due volte l'algoritmo del simplesso.

1. Introduzione

1.1 Problemi di soddisfacibilità

Il problema della soddisfacibilità di variabili booleane all'interno di una espressione contenente operatori booleani è il ben noto problema della soddisfacibilità booleana.
(Soddisfacibilità booleana - Wikipedia)
La domanda è la seguente: data un'espressione, esiste un qualche assegnamento di valori TRUE e FALSE tali da rendere l'intera espressione vera? Una formula di logica proposizionale è detta soddisfacibile se esiste tale assegnamento.

Nel caso generale un problema di soddisfacibilità è descritto da un sistema di disequazioni lineari non strette del tipo $\mathbf{Ax} \leq \mathbf{b}$. In questo caso il problema posto è se esiste o meno un assegnamento per la variabile **x** che soddisfi tale sistema.
Quest'ultimo tipo di problemi costituiscono la base dei problemi e modelli di programmazione lineare, ampiamente trattati nel libro di Ricerca Operativa del professore Pappalardo [1]. Esistono già diversi modi per risolvere tali problemi, uno dei più famosi è l'algoritmo del simplesso [2,3] di Dantzig.

1.2 Descrizione del problema

Nel nostro caso invece il problema che ci poniamo è se esiste o meno un assegnamento per la variabile x in R^n che soddisfi il seguente sistema di disequazioni lineari strette: $\mathbf{A_s x < b_s}$.
L'introduzione di questi nuovi vincoli altera significativamente il modello, dunque la trattazione del problema. Le differenze e le implicazioni dovute a ciò verranno trattate nel seguente capitolo.

2. Implicazioni dovute ai vincoli descritti con disequazioni lineari strette

2.1 Differenze con i classici problemi di Linear programming

Le differenze rispetto al caso precedente potrebbero sembrare minime ma l'uso di disequazioni lineari strette ha implicazioni notevoli nella trattazione del problema.
Ad esempio consideriamo il problema di Linear Programming (P1):

$$max\ x_1 + x_2$$
$$-2x_1 + x_2 \leq 1 \quad (1)$$
$$x_1 - 2x_2 \leq -4 \quad (2)$$
$$x_1 + x_2 \leq 14 \quad (3)$$
$$-x_1 - x_2 \leq -14 \quad (4)$$

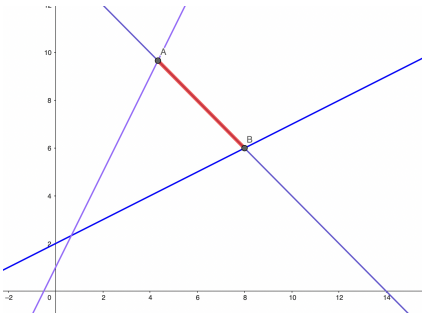


fig.1 Dominio del problema P1 (segmento AB)

Dalla **fig.1** notiamo come il dominio, nonché l'ottimo, del problema P1 sia costituito dall'intero segmento AB, dunque avremo infiniti punti ottimi. In particolare questo comportamento è dovuto alla presenza dei vincoli 3 e 4.

Invece trattando lo stesso problema, utilizzando il modello di nostro interesse (problema P2), dunque utilizzando vincoli di minore stretto, avremo tutt'altro risultato.

$$max\ x_1 + x_2$$
$$-2x_1 + x_2 < 1$$
$$x_1 - 2x_2 < -4$$
$$x_1 + x_2 < 14$$
$$-x_1 - x_2 < -14$$

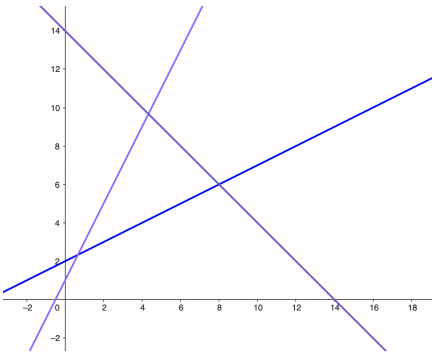


fig.2 Dominio per problema P2 (l'insieme vuoto)

In questo caso il dominio del nostro problema è l'insieme vuoto. Quest'ultimo è un semplice esempio, nel quale possiamo notare chiaramente la differenza tra i due problemi. Dunque è ancor più chiaro che non possiamo trattarli nello stesso modo. Lo scopo di questa tesi è di trovare una tecnica risolutiva per i problemi di tipo P2, utilizzando concetti già e a noi familiari, cioè algoritmi e tecniche già utilizzati per risolvere i problemi di tipo P1. In particolare utilizzeremo un approccio che fa riferimento all'algoritmo del simplesso.

2.2 Prima soluzione con parametri η

Una prima soluzione potrebbe essere data dall'introduzione di un parametro molto piccolo, detto η, supposto inizialmente uguale a 10^{-3}. Il nostro scopo è quello di utilizzare questo parametro per trasformare il nostro problema di tipo P2 in uno di tipo P1 (cioè con vincoli di ≤), d'ora in poi questo problema trasformato verrà chiamato P3. In modo tale da poter usare gli algoritmi già esistenti per la risoluzione di problemi di PL anche per il nostro problema.

$$\textbf{P2:}\quad \min cx,\ \mathbf{A_s x < b_s} \quad \Longrightarrow \quad \textbf{P3:}\quad \min cx,\ \mathbf{A_s x \leq b_s - \eta}$$

Il problema di questa soluzione, oltre al fatto che η non è arbitrario, è il fatto che a seconda di come scegliamo il parametro η potremmo o meno trovare un insieme di soluzioni per il nostro problema. Osserviamo il seguente esempio:
Introduciamo il parametro η nel sistema precedente:

$$\begin{aligned}
&-2x_1 + x_2 \leq 1 - 10^{-3}\\
&x_1 - 2x_2 \leq -4 - 10^{-3}\\
&x_1 + x_2 \leq 14 - 10^{-3}\\
&-x_1 - x_2 \leq -14 - 10^{-3}
\end{aligned}$$

Ciò che ci auspichiamo è di ottenere un insieme di soluzione vuoto come nel caso P2 precedente.

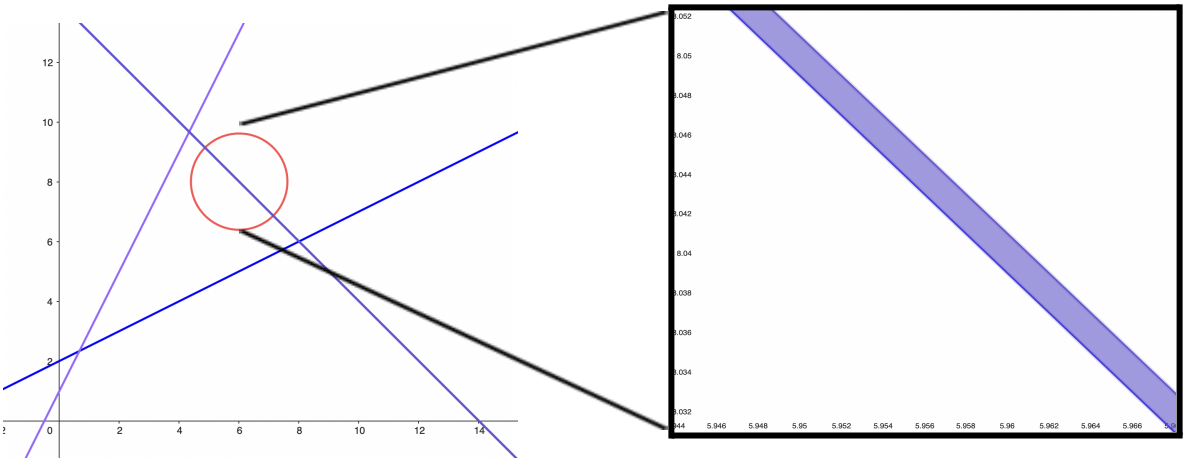


fig.3 Dominio del problema dopo introduzione parametro η, con annesso zoom

Purtroppo la situazione è ben diversa da quella augurata. Come si può notare dalla **fig.3,** l'insieme dei punti del dominio non è vuoto, bensì è un'intera area.

2.3 Soluzione con numeri infinitesimi

Per risolvere questo problema abbiamo bisogno di introdurre il concetto di numero infinitesimo, ripreso dalla α-Theory [4]. Detto α, un numero infinito ben preciso, avremo che a $\eta = \frac{1}{\alpha}$ corrisponde ad un numero infinitesimo, infinitamente piccolo. Trattandosi di numeri sarà possibile continuare ad eseguire operazioni con essi.

Grazie all'introduzione del concetto di numero infinitesimo, riusciamo a risolvere i problemi del sistema introdotto nel paragrafo precedente (P3), però ora la trattazione del problema necessità di ulteriori approfondimenti e osservazioni.

Difatti il nuovo sistema, ora, ha una particolarità: il termine noto delle disequazioni del sistema non è più un comune termine noto, bensì si dirà che il termine è Non-Archimedeo. Con il termine Non-Archimedeo si intende un uso di numeri che non sono solo finiti ma che possono essere anche infiniti o infinitesimi.

Dunque si tratta di una generalizzazione, questo tipo di problemi è detto *Linear Programming with Non-Archimedean Right-Hand Sides* (LPNARHS). Questa generalizzazione è stata trattata nel recente articolo di ricerca *Linear Programming with Non-Archimedean Right-Hand Sides* [5] del professor Cococcioni. Nel quale viene introdotto il problema e viene proposta una tecnica risolutiva mediante un algoritmo G-RHS-SIMPLEX, una generalizzazione del simplesso di Dantzig [2] , che funziona anche in caso di termini Non-Archimedei.

Questo algoritmo è basato sulla teoria Grossone-based Methodology [6,7] di Sergeyev, che introduce un nuovo concetto di infinito, rappresentato dal simbolo ①, tale simbolo è detto grossone e corrisponde ad un numero naturale (dunque gode di tutte proprietà dei numeri naturali) maggiore di tutti i numeri naturali finiti.

La ricerca [5] è ispirata da un altro lavoro [8], nel quale viene trattato una classe di problemi molto importanti: *Lexicographic Multi-Objective Linear Programming* (LMOLP). Una classe di problemi nei quali sono presenti funzioni multi-obiettivo e in cui ciascun obiettivo ha un livello di importanza maggiore del successivo.
Questi due approcci verranno approfonditi nel capitolo successivo.

3. Dal problema Primale al problema Duale Lessicografico

Facciamo un passo indietro, ricapitolando i passaggi fatti fino ad ora.
Inizialmente il nostro scopo era di risolvere il problema dato da un sistema di disequazioni lineari strette nella forma:

$$\min cx$$
$$\mathbf{A_s x} < \mathbf{b_s}$$

Dopodiché, dato che volevamo risolvere il problema utilizzando un algoritmo già molto conosciuto e diffuso come il simplesso, che però risolve solo problemi con vincoli lineari non stretti. Abbiamo introdotto un parametro η, prima finito, ma ciò comportava diverse problematiche, poi infinitesimo; per ottenere un problema nella forma:

$$\min cx$$
$$\mathbf{A_s x} \leq \mathbf{b_s} - \eta$$

Questo tipo di problema fa parte della classe dei LPNARHS, i quali non possono essere risolti da un semplice simplesso, in quanto utilizzano termini che possono essere sia finiti che infiniti o infinitesimi

A questo punto ricorriamo alla teoria della dualità, trattata anch'essa in [1]. La teoria della dualità ci dice che risolvere il problema precedente (P3), detto primale, equivale a risolvere il suo duale, cioè:

$$\max (\mathbf{b_s} - \eta)^T y$$
$$y \mathbf{A_s} = c$$
$$y \geq 0$$

In questo modo otteniamo un problema del tipo Dual Lexicographic (DLEX). Per la risoluzione di questa classe di problemi possiamo usare una tecnica risolutiva simile a quella proposta per i problemi del tipo *Lexicographic multi-objective linear programming* in [8]. Questa classe di problemi si presentano nella seguente forma:

$$\text{LexMax } c^1x, c^2x, \ldots, c^rx$$
$$\text{s.t. } \{x \in R^n : Ax = b\,,\ x \geq 0\} \qquad \textbf{(P4)}$$

Nel nostro caso il problema sarà dunque nella forma:

LexMax $\mathbf{b_s}^T y$, $\mathbf{-1}^T y$ (da notare che non è più presente η)

s.t. $\{y \in R^n: y^T \mathbf{A_s} = c , y \geq 0\}$

In questa forma vengono presentati problemi multi-obiettivo, cioè problemi con più funzioni obiettivo, ognuno con importanza maggiore del successivo.
Dunque avremo che c^1x è più importante di c^2x e così via.

Ma come facciamo a discriminare le funzioni in base alla loro importanza?
Esistono due soluzioni a questa domanda, una per ogni approccio risolutivo che vogliamo adottare: il modo preemptive e il modo non preemptive.

3.1 Risoluzione preemptive

Questa tecnica risolve il problema in modo iterativo, cioè risolve il problema LP multi-obiettivo come una serie di problemi LP a singolo obiettivo, iniziando dal primo, il più importante, e si ferma quando ha risolto l'ultimo.

max c_1x

s.t. $\{x \in R^n: Ax = b , x \geq 0\}$

Dato che il problema a singolo-obiettivo è ben noto, lo possiamo risolvere utilizzando un normale algoritmo del simplesso.
Un particolare da notare però è il fatto che dalla seconda iterazione, dunque anche alle iterazioni successive, venga aggiunto un vincolo in più, corrispondente all'ottimo della funzione obiettivo precedentemente eseguita.

max c_2x

s.t. $\{x \in R^n: Ax = b , x \geq 0, \mathbf{c_1}^T\mathbf{x} = \mathbf{c_1}^T\mathbf{x_{opt1}}=\beta_1 \}$

L'algoritmo si fermerà alla *r*-esima iterazione, dove *r* è l'indice dell'ultima funzione obiettivo del nostro problema multi-obiettivo. Chiaramente questo approccio ha un costo computazionale molto elevato, ma vedremo che nel nostro caso, in cui abbiamo solo due iterazioni, questo approccio può essere adeguato a risolvere il problema.

3.2 Risoluzione non preemptive

Questa strategia risolutiva prevede invece l'uso di un algoritmo detto Gross-Simplex, basato sulla Grossone-based Methodology [6,7]. In sostanza, grazie all'introduzione del nuovo concetto di grossone, il problema multi-obiettivo P4, può essere scritto come un problema gross-LP nella forma:

$$\max c_1x + c_2x \cdot \eta + c_3x \cdot \eta^2 + \ldots\ldots + c_rx \cdot \eta^{r-1}$$
$$Ax = b$$
$$x \geq 0 \qquad \textbf{(P5)}$$

Dove $\eta = \frac{1}{\alpha}$

Il problema presentato in questa forma può essere risolto dall'algoritmo gross-simplex presentato in [8]. Come si può notare il problema che prima presentava molteplici funzioni obiettivo, di importanza diversa, ora è un problema con una singola funzione obiettivo, ma che presenta sia termini finiti che infinitesimi. Quest'ultimo particolare non è affatto banale, anzi, rende la risoluzione del problema computazionalmente molto più costoso.

4. Il problema DLEX

Dalle considerazioni fatte nel capitolo precedente osserviamo che il nostro problema di tipo DLEX possa essere considerato come un problema di tipo P5 e dunque possa essere risolto mediante le stesse metodologie. In particolare tenteremo di risolvere il problema con un approccio preemptive dato che le variabili della funzione obiettivo sono solamente due, dunque bastano due iterazioni del metodo preemptive per risolvere il problema.

4.1 Cosa andremo a fare?

Partiamo dal nostro problema DLEX in forma:

$$\max (\mathbf{b_s} - \eta)^T y$$
$$y \mathbf{A_s} = c$$
$$y \geq 0$$

Primo step: trovare l'ottimo del seguente problema utilizzando un normale algoritmo del simplesso:

$$\max \mathbf{b_s}^T y$$
$$y \mathbf{A_s} = c$$
$$y \geq 0$$

Secondo step: sia $\mathbf{b_s} \cdot y_{opt1} = \beta_1$ l'ottimo della funzione obiettivo trovata al punto precedente, imponiamo tale assegnamento come ulteriore vincolo nell'iterazione successiva e risolviamo il problema successivo seguente:

$$\max \mathbf{(-1)^T} y$$
$$y \mathbf{A_s} = c$$
$$\mathbf{b_s} \cdot y \geq \beta_1$$
$$y \geq 0$$

In conclusione, l'ottimo così trovato corrisponderà alla soluzione ottima del nostro problema iniziale, e dunque avremo raggiunto il nostro obiettivo.
Come già accennato precedentemente, notiamo come il problema P2 che ci eravamo posti all'inizio può essere risolto iterativamente usando due volte un semplice algoritmo del simplesso, ovviamente introducendo alcune accortezze.

Prima di continuare, è importante fare un osservazione: il nostro problema nella forma DLEX ma con una funzione obiettivo Non-Archimedea è stata scomposta in due problemi, da eseguire iterativamente (uno dopo l'altro in ordine), questa volta con funzione obiettivo finita, risolvibili con un semplice algoritmo del simplesso.

Da notare, in particolare, come il numero infinitesimo η, introdotto nel problema, grazie ad una risoluzione preemptive

4.2 Esempio con un caso degenere

Per comprendere ancora meglio il problema, prendiamo in considerazione un caso con soluzione degenere.

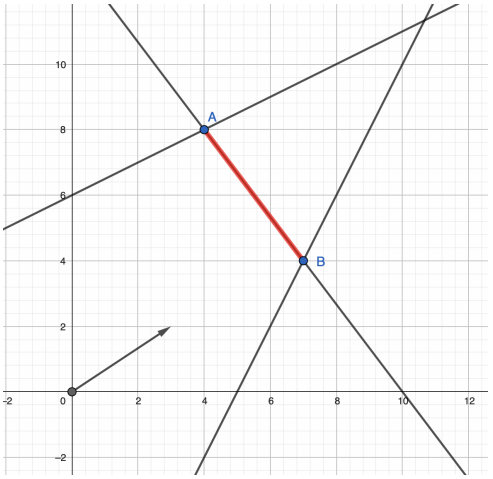


fig.4 Caso di ottimo degenere

Supponiamo che dopo aver utilizzato un algoritmo del simplesso dopo lo step1, ci troviamo nel caso in cui abbiamo due basi ottime e dunque la soluzione ottima del nostro problema è costituita dall'insieme dei punti sul segmento che unisce le due basi ottime. (**fig. 4**)

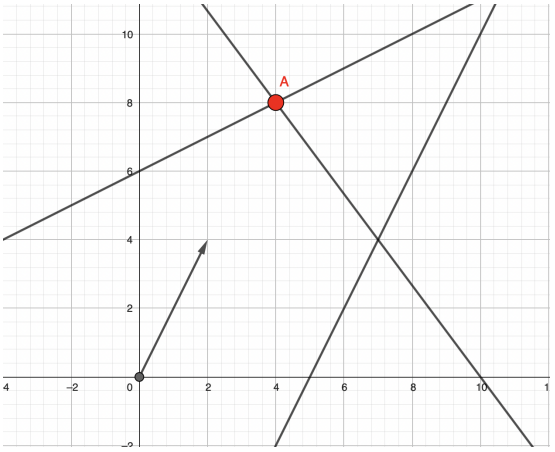


fig.5 Soluzione ottimo dopo lo step2

In questo caso dopo step2 otterremo un risultato diverso rispetto allo step1, ovviamente, perchè è cambiata la funzione obiettivo. Dopo aver eseguito nuovamente il simplesso questa volta otteniamo che l'ottimo si trova su un vertice del poliedro.

Da questo semplice esempio si osserva come il nostro problema iniziale sia molto più complesso e risolvendolo nella forma DLEX con un approccio preemptive a due iterazioni otteniamo un risultato completamento diverso da un semplice problema di Linear Programming. Nei seguenti capitoli verranno trattati alcuni esempi di problemi P2.

5. Risoluzione con approccio preemptive

In questo capitolo useremo l'approccio preemptive, mostrato precedentemente, per la risoluzione di alcuni problemi (alcuni già trattati in questa tesi e altri nuovi).
In particolare entreremo più nel dettaglio mostrando alcuni problemi risolti utilizzando questo approccio. Risolveremo tali problemi utilizzando la funzione di matlab: *Linprog*.

Prima di iniziare con i vari esempi, riassumiamo l'algoritmo che useremo per applicare l'approccio preemptive ai vari problemi.

Algoritmo Approccio Preemptive

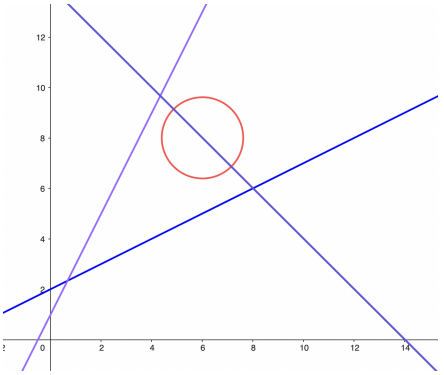
0: Dato un problema $Ax < b$ lo porto in forma $Ax \leq b - \eta$
1: Lo trasformiamo nella forma DLEX
2: Scomponiamo il problema in due problemi da risolvere sequenzialmente,
il primo costituito dagli stessi vincoli e con funzione obiettivo costituito solo dalla parte finita dei coefficienti della funzione obiettivo iniziale, il secondo con gli stessi vincoli + un vincolo pari all'ottimo trovato nel problema precedente e con funzione obiettivo costituito dalla parte infinitesima dei coefficienti della funzione obiettivo iniziale
3: Risolvo il primo problema utilizzando linprog e trovando così y_{opt1} e il corrispondente $f_{opt1} = b_1^T \cdot y_{opt1}$ dove b_1^T è la parte finita
4: Risolvo il secondo problema, con il vincolo aggiuntivo, utilizzando linprog e trovo y_{opt2} e il corrispondente $f_{opt2} = b_2^T \cdot y_{opt2}$ dove b_2^T è la parte infinitesima
5: La soluzione ottima del problema nel complesso è $y_{opt} = y_{opt2}$ e il valore della funzione obiettivo è pari a $f_{opt} = b^T \cdot y_{opt2}$ dove b^T è costituita sia dalla parte infinitesima che dalla parte finita

---

5.1 Discriminare un problema con regione ammissibile vuota da una con regione ammissibile

Trattiamo nuovamente un problema già posto precedentemente:

$$\max\ 3x_1+3x_2$$
$$-2x_1+x_2< 1$$
$$x_1-2x_2< -4$$
$$x_1+x_2< 14$$
$$-x_1-x_2< -14$$



problema che sappiamo non avere soluzione (esempio precedente), dunque ci aspettiamo che con l'approccio preemptive ci dia come risultato l'insieme vuoto.

$$\max\ 3x_1+3x_2$$
$$-2x_1+x_2 \leq 1-\eta$$
$$x_1-2x_2 \leq -4-\eta$$
$$x_1+x_2 \leq 14-\eta$$
$$-x_1-x_2 \leq -14-\eta$$

portiamo il problema nella forma DLEX

$$\min\ (1-\eta)\,y_1+(-4-\eta)\,y_2+(14-\eta)\,y_3+(-14-\eta)\,y_4$$
$$-2y_1+y_2+y_3-y_4=3$$
$$y_1-2y_2+y_3-y_4=3$$
$$y_i \geq 0$$

adotto ora l'approccio preemptive per la risoluzione del problema DLEX

$$\min\ y_1-4y_2+14\,y_3-14\,y_4$$
$$-2y_1+y_2+y_3-y_4=3$$
$$y_1-2y_2+y_3-y_4=3$$
$$y_i \geq 0$$

La funzione linprog(f, A,b,Aeq, beq) di matlab risolve il problema.
L'ottimo della funzione obiettivo così trovato risulta valere $f_{opt1}= 42$
trovato per $y_{opt1}=[0, 0, 3, 0]$

In questo caso, una soluzione ammissibile è più che plausibile dato che la tipologia dei vincoli è cambiata, la discriminazione verrà effettuata allo step successivo.

Al secondo step , risolvo il seguente problema utilizzando sempre linprog.

$\min \; -y_1 - y_2 - y_3 - y_4$

$-2y_1 + y_2 + y_3 - y_4 = 3$
$y_1 - 2y_2 + y_3 - y_4 = 3$
$y_1 - 4y_2 + 14\, y_3 - 14\, y_4 = 42$ // NB: con l'uguale funziona mentre con <= no
$y_i \geq 0$

Il motivo per cui al secondo step è stato aggiunto un vincolo è il seguente:
al primo step abbiamo trovato un valore della funzione obiettivo corrispondente f_{opt1} ,il quale deve essere comunque ancora rispettato perché la funzione obiettivo precedente ha priorità maggiore rispetto a quella del secondo problema. Dunque la nuova y, trovata allo step 2, deve rispettare obbligatoriamente questo vincolo.
In questo caso, linprog restituisce: "Problem is unbounded", dunque come ci aspettavamo non esiste una soluzione finita per il problema.

Per dimostrare che il nostro approccio funziona, devo dimostrare che questa tecnica risolutiva riesca a differenziare i casi limite proposti precedentemente. Dunque, in questo caso deve riuscire a discriminare se un problema ha soluzione o meno.
Per tale motivo adottiamo lo stesso approccio anche al seguente problema, aspettandoci che questa volta linprog dia una soluzione ammissibile.

$\max \; 3x_1 + 3x_2$

$-2x_1 + x_2 < 1$
$x_1 - 2x_2 < -4$
$x_1 + x_2 < 14$

Applicando le stesse trasformazioni fatte in precedenza ottengo:

$\min (1-\eta)\, y_1 + (-4-\eta)\, y_2 + (14-\eta)\, y_3$
$-2y_1 + y_2 + y_3 = 3$
$y_1 - 2y_2 + y_3 = 3$
$y_i \geq 0$

adotto la tecnica preemptive nuovamente e risolvo il seguente problema:

$$\begin{aligned} \min\ & y_1 - 4y_2 + 14\, y_3 \\ & -2y_1 + y_2 + y_3 = 3 \\ & y_1 - 2y_2 + y_3 = 3 \\ & y_i \geq 0 \end{aligned}$$

$f_{opt1} = 42$ trovato per $y_{opt1} = [0, 0, 3]$
allo step successivo risolvo:

$$\begin{aligned} \min\ & -y_1 - y_2 - y_3 \\ & -2y_1 + y_2 + y_3 = 3 \\ & y_1 - 2y_2 + y_3 = 3 \\ & y_1 - 4y_2 + 14\, y_3 = 42 \\ & y_i \geq 0 \end{aligned}$$

Questa volta il problema ha soluzione, pari a $f_{opt2} = 42$ trovato per $y_{opt2} = [0, 0, 3]$

Come speravamo, adottando la tecnica preemptive, siamo riusciti grazie a questo semplice esempio a dimostrare che un problema complesso di tipo $Ax < b$, in quale non è direttamente risolvibile usando un semplice algoritmo come il simplesso, si possa risolvere adottando un approccio preemptive.
In particolare in questo caso siamo riusciti a risolvere uno dei problemi principali che ci eravamo posti inizialmente: una delle soluzioni banali proposti inizialmente era quella di introdurre un parametro molto piccolo, ma come dimostrato precedente questa soluzione non era affatto soddisfacente proprio perché non riusciva a discriminare problemi con insieme soluzione vuoto.
Mentre da questo esempio abbiamo dimostrato che l'approccio preemptive riesce a distinguere i due casi. Quest'ultimo caso verrà ripreso e approfondito del successivo capitolo.

5.2. Risoluzione di un problema già risolto con Gross-Simplex

In questo paragrafo andremo a usare l'approccio preemptive per risolvere il problema 1 trattato in [8], questo problema è stato già risolto usando un approccio non preemptive, in particolare usando un algoritmo Gross-simplex. Il problema in questione aveva 3 funzioni obiettivo diverse, noi ci limiteremo a due.
Lo scopo dunque sarà quello di dimostrare che entrambi gli approcci sono corretti e portano allo stesso risultato.
Il problema in questione è il seguente:

lexmax $8x_1 + 12x_2, 14x_1 + 10x_2$

s.t. $2x_1 + x_2 + x_3 = 120$,

$2x_1 + 3x_2 + x_4 = 210$,

$4x_1 + 3x_2 + x_5 = 270$,

$-x_1 + 2x_2 + x_6 = -60$,

$x1,x2,x3,x4,x5,x6 \geq 0$ **(P6)**

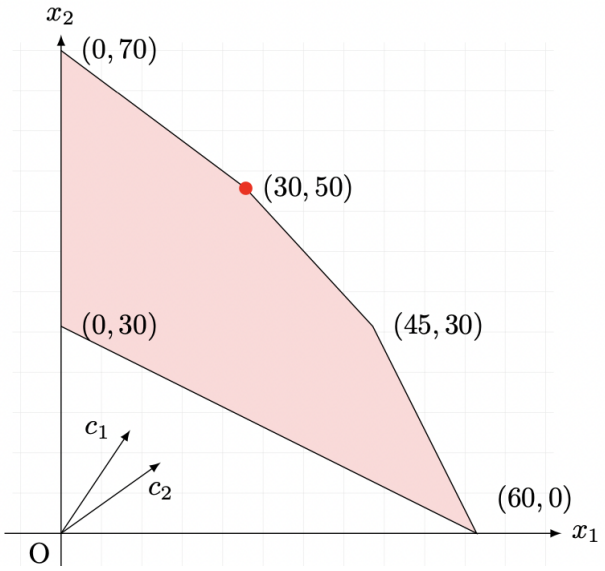


fig.6 Rappresentazione grafica del problema (P6): c1 sta per la prima funzione obiettivo e c2 per la seconda

Il valore della funzione obiettivo e la corrispondente soluzione ottima trovata dall'algoritmo Gross-simplex è il seguente: y_{opt}=[30; 50; 10; 0; 0; 70] e
$f_{opt} = b^T y_{opt} = 840 + 920\eta$

Ora utilizzando l'approccio preemptive ci auguriamo di trovare lo stesso risultato.
Il problema P6 può essere scritto come:

$$\min -(8 + 14\eta)\, x_1 - (12 + 10\eta)x_2$$
$$2x_1 + x_2 + x_3 = 120$$
$$2x_1 + 3x_2 + x_4 = 210$$
$$4x_1 + 3x_2 + x_5 = 270$$
$$-x_1 + 2x_2 + x_6 = -60$$
$$x1,x2,x3,x4,x5,x6 \geq 0$$

*il segno è stato cambiato perchè lo abbiamo trasformato in un problema di minimo

Seguendo l'approccio preemptive, risolvo prima il problema:

$$\min -8\,x_1 - 12x_2$$
$$2x_1 + x_2 + x_3 = 120$$
$$2x_1 + 3x_2 + x_4 = 210$$
$$4x_1 + 3x_2 + x_5 = 270$$
$$-x_1 - 2x_2 + x_6 = -60$$
$$x1,x2,x3,x4,x5,x6 \geq 0$$

Linprog restituisce soluzione ottima y_{opt1} =[0; 70; 50; 0; 60; 80] e f_{opt1}= -840
Allo step successivo il problema da risolvere è:

$$\min -14x_1 - 10x_2$$
$$2x_1 + x_2 + x_3 = 120$$
$$2x_1 + 3x_2 + x_4 = 210$$
$$4x_1 + 3x_2 + x_5 = 270$$
$$-x_1 - 2x_2 + x_6 = -60$$
$$-8\,x_1 - 12x_2 = -840$$
$$x1,x2,x3,x4,x5,x6 \geq 0$$

La soluzione ottima trovata da linprog è y_{opt2}=[30; 50; 10; 0; 0; 70] e f_{opt2}= -920
Eliminando le variabili di scarto abbiamo che la soluzione ottima al problema rappresentato in **fig.6** è x=[30;50] come ci auguravamo. Questo è l'esito che speravamo. Siamo riusciti a giungere alla stessa soluzione ottima adottando però un approccio preemptive, che in questo caso dato che le iterazioni da fare sono solo 2, è molto più efficiente e ha un costo computazionale minore.

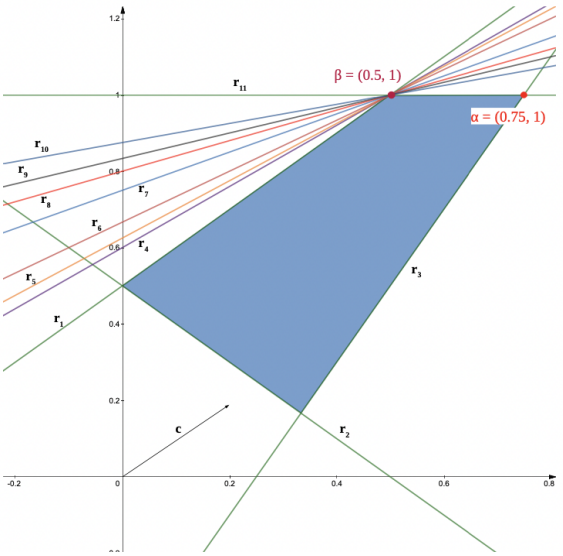
Questo problema è un classico problema con caso degenere (presentato nel capitolo precedente), nel quale dopo la prima iterazione (dove risolviamo il problema con la prima funzione obiettivo) troviamo un segmento di punti ottimi e successivamente alla seconda iterazione (con la seconda funzione obiettivo) "miglioriamo" la soluzione, trovando un punto preciso corrispondente all'ottimo del nostro problema nel complesso.

5.3. Un caso fortemente degenere

Per approfondire ancor meglio i problemi degeneri, trattiamo in questo paragrafo un problema, tratto da [5], nel quale si presenta un problema con un dominio che presenta un vertice altamente degenere nel punto $\beta = [0.5 , 1]$.

$$\max x_1+x_2$$

$$\begin{aligned}
-2x_1 + 2x_2 &\leq 1\\
-2x_1 - 2x_2 &\leq 1\\
4x_1 - 2x_2 &\leq 1\\
-4x_1 + 5x_2 &\leq 3\\
-6x_1 + 8x_2 &\leq 5\\
-2x_1 + 3x_2 &\leq 2\\
-2x_1 + 4x_2 &\leq 3\\
-2x_1 + 5x_2 &\leq 4\\
-2x_1 + 6x_2 &\leq 5\\
-2x_1 + 8x_2 &\leq 7\\
x_2 &\leq 1
\end{aligned}$$



Per rimuovere la degenerazione è stato introdotto un vettore rumore **w** infinitesimo, generato casualmente e aggiunto al vettore **b**. In questo modo si riesce a rimuovere la degenerazione, il problema ottenuto ora è un problema di programmazione lineare non-Archimedeo non degenere.

$$\max x_1+x_2$$

$$\begin{aligned}
-2x_1 + 2x_2 &\leq 1 + 0.17\eta\\
-2x_1 - 2x_2 &\leq 1 + 0.2\eta\\
4x_1 - 2x_2 &\leq 1 + 0.23\eta\\
-4x_1 + 5x_2 &\leq 3 + 0.27\eta\\
-6x_1 + 8x_2 &\leq 5 + 0.32\eta\\
-2x_1 + 3x_2 &\leq 2 + 0.38\eta\\
-2x_1 + 4x_2 &\leq 3 + 0.44\eta\\
-2x_1 + 5x_2 &\leq 4 + 0.52\eta\\
-2x_1 + 6x_2 &\leq 5 + 0.61\eta\\
-2x_1 + 8x_2 &\leq 7 + 0.72\eta\\
x_2 &\leq 1 + 0.85\eta
\end{aligned}$$

Questo problema, come ben sappiamo, può essere trasformato in forma DLEX:

$$\begin{aligned}
\min\ & (1 + 0.17\eta)\, y_1 + (1 + 0.2\eta)\, y_2 + (1 + 0.23\eta)\, y_3 + (3 + 0.27\eta)\, y_4 +\\
& + (5 + 0.32\eta)\, y_5 + (2 + 0.38\eta)\, y_6 + (3 + 0.44\eta)\, y_7 +\\
& + (4 + 0.52\eta)\, y_8 + (5 + 0.61\eta)\, y_9 + (7 + 0.72\eta)\, y_{10} + (1 + 0.85\eta)\, y_{11}
\end{aligned}$$

$$2y_1 - 2y_2 + 4y_3 - 4y_4 - 6y_5 - 2y_6 - 2y_7 - 2y_8 - 2y_9 - 2y_{10} = 1$$

$$2y_1 -2y_2 -2y_3 + 5y_4 + 8y_5+3y_6+4y_7+5y_8+6y_9+8y_{10}+y_{11}=1$$
$$y_i \geq 0$$

Adottiamo nuovamente l'approccio preemptive per la risoluzione del problema. Dunque al primo step risolvo:

$$\min\ y_1 + y_2 + y_3 + 3y_4 + 5y_5 + 2y_6 + 3y_7 + 4y_8 + 5y_9 + 7y_{10} + y_{11}$$

$$-2y_1 -2y_2 + 4y_3 -4y_4 -6y_5-2y_6-2y_7-2y_8-2y_9-2y_{10}= 1$$
$$2y_1 -2y_2 -2y_3 + 5y_4 + 8y_5+3y_6+4y_7+5y_8+6y_9+8y_{10}+y_{11}=1$$
$$y_i \geq 0$$

La soluzione ottima proposta da linprog è:
y_{opt1} =[0, 0, 0.25, 0, 0 ,0 ,0 ,0 ,0 1.5] e f_{opt1} = 1.75

Al secondo step risolvo:

$$\min 0.17\ y_1 + 0.2y_2 + 0.23y_3 + 0.27y_4 + 0.32\ y_5 + 0.38\ y_6 + 0.44y_7 + 0.52y_8 + 0.61y_9 + 0.72y_{10} + 0.85y_{11}$$

$$-2y_1 -2y_2 + 4y_3 -4y_4 -6y_5-2y_6-2y_7-2y_8-2y_9-2y_{10}= 1$$
$$2y_1 -2y_2 -2y_3 + 5y_4 + 8y_5+3y_6+4y_7+5y_8+6y_9+8y_{10}+y_{11}=1$$
$$y_1 + y_2 + y_3 + 3y_4 + 5y_5 + 2y_6 + 3y_7 + 4y_8 + 5y_9 + 7y_{10} + y_{11} = 1.75$$
$$y_i \geq 0$$

La soluzione ottima al secondo step di y rimane la stessa e la f_{opt2}=1.3325
Dunque il valore ottimo della nostra funzione obiettivo nel complesso è
f_{opt}= 1.75 + 1.3325η, medesimo risultato trovato in [5].

5.4. Vantaggi dell'approccio preemptive

Con questi esempi abbiamo dimostrato che l'approccio preemptive non solo è corretto, portando al medesimo risultato dell'approccio non preemptive con uso di un algoritmo gross-Simplex, trattato negli articoli [5] e [8], ma nei casi in cui abbiamo funzioni obiettivo con una parte finita e solo una parte infinitesima, è addirittura un'alternativa migliore dato che ha un costo computazionale inferiore.
Inoltre grazie a questo approccio possiamo risolvere problemi di programmazione lineare non-Archimedei, utilizzando semplicemente una funzionalità di matlab, il linprog, il quale è un risolutore molto conosciuto e soprattutto fortemente ottimizzato.
Uno dei più grandi vantaggi di questo approccio è che a differenza dell'algoritmo Gross-simplex, con l'approccio preemptive non lavoriamo mai direttamente con i numeri infinitesimi, infatti ad ogni step dell'algoritmo

lavoriamo solo con numeri finiti. Nella funzione obiettivo iniziale (non-Archimedea) viene distinta la parte finita da quella infinitesima, la prima andrà a costituire la funzione obiettivo del problema al primo step nonché i coefficienti del vincolo aggiuntivo al secondo step, mentre la parte infinitesima costituirà la funzione obiettivo del problema al secondo step. Quest'ultimo problema però non avrà una funzione obiettivo infinitesima, bensì sarà costituito da soli numeri finiti. Possiamo adottare questa semplificazione a patto che la prima funzione obiettivo mantenga una priorità maggiore rispetto alla seconda, e ciò è garantito dal vincolo aggiuntivo introdotto nel secondo step.

Infine, tornando allo scopo iniziale di questa trattazione, utilizzando questa tecnica risolutiva possiamo andare a risolvere il problema di tipo $Ax < b$ (con vincoli di minore stretto), la quale trattazione non è per nulla banale, in due semplici passaggi usando un risolutore, come anticipato precedentemente, fortemente ottimizzato: il linprog di matlab. Problemi in questa forma trovano ampia applicazione nei **Theorem Provers e nei tool di Formal Method Verification.**

6. RIEPILOGO

6.1 Algoritmo risolutivo definitivo

In questo ultimo capitolo introduciamo un algoritmo definitivo per la risoluzione del problema iniziale, in forma **Ax < b.** In particolare ci interessa capire se un determinato problema in quella forma è feasible o meno, cioè se ammette soluzione o meno.
Tale algoritmo può essere schematizzato nel seguente modo:

Algoritmo Per la risoluzione del problema in forma Ax < b

0: Dato un problema Ax < b risolvo il corrispondente problema Ax ≤ b con un normale algoritmo del simplesso.

1: Se non ha soluzione, concludo che il nostro problema non ammette soluzione.

2: Altrimenti, procedo con la risoluzione mediante approccio Preemptive.

3: Se al termine della seconda iterazione, il problema presenta soluzione ammissibile allora il problema è compatibile.

D'ora in poi questo algoritmo sarà definito "algoritmo definitivo"

In sostanza, prima di risolvere il problema Ax < b con l'algoritmo presentato nel capitolo 5, risolvo il corrispondente problema nella forma Ax ≤ b. Se quest'ultimo non ammette soluzione posso concludere che neanche il problema iniziale ha soluzioni ammissibili (Problema Chiaramente Incompatibile). Ciò perché l'insieme delle soluzioni ammissibili del primo problema è incluso nell'insieme delle soluzioni ammissibili del secondo. Se invece ammette soluzioni non posso concludere che anche il problema iniziale sia risolubile (Problema apparentemente compatibile). Devo necessariamente risolvere il problema con l'approccio preemptive presentato nel capitolo precedente. In particolare se alla seconda iterazione dell'algoritmo restituisce una soluzione bounded, potremmo concludere che il Problema iniziale è compatibile e ammette soluzioni, altrimenti se presenta come risultato "Problem Unbounded" posso concludere definitivamente che il problema è Incompatibile.

In fig.7, viene mostrato lo schema riassuntivo dell'algoritmo utilizzato.

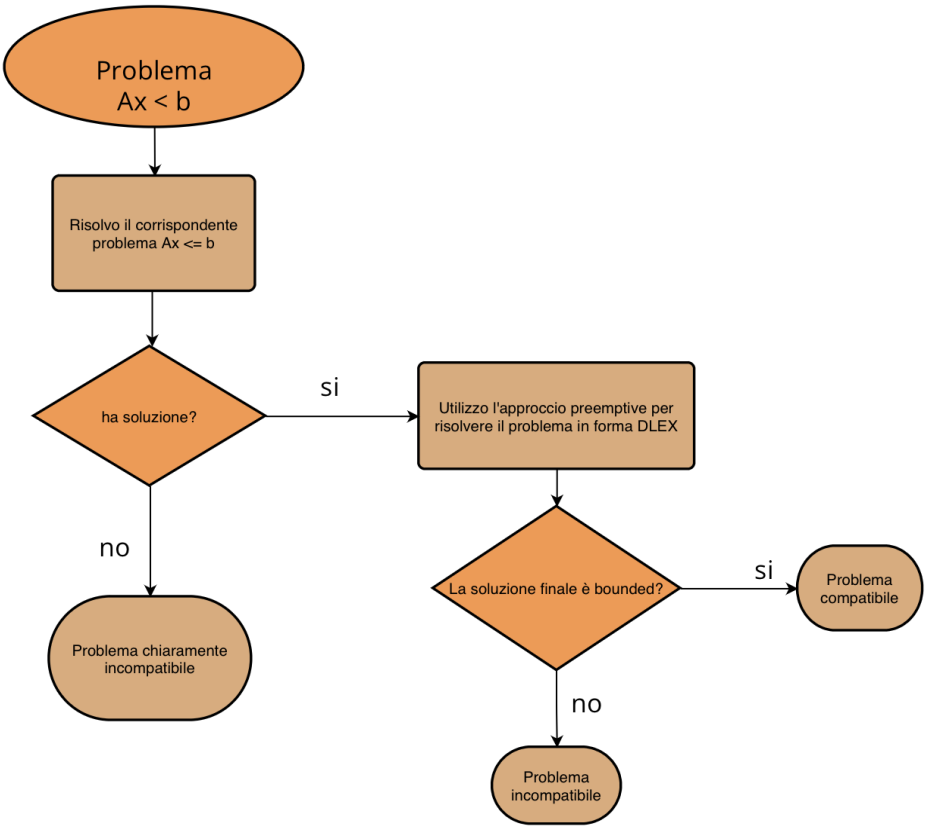
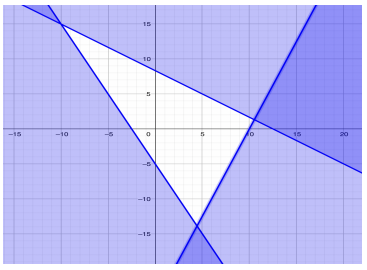


fig.7 Schema riassuntivo dell'algoritmo utilizzato

Con questa semplice modifica riusciamo a scartare a priori i casi in cui il nostro problema già nella prima fase, cioè quando è in forma **Ax ≤ b** , non presenta alcuna soluzione ammissibile. Dunque possiamo già affermare che il problema **Ax < b** non ammette soluzioni ammissibili. Per chiarire le idee, ecco un esempio:

1. Dato il problema:

$$\max \; 3x_1+3x_2$$
$$-3x_1- x_2< 25$$
$$2x_1+ x_2< 5$$
$$-5x_1 + 2x_2< 50$$



2. Risolvo il problema nella forma Ax ≤ b

$$\max\ 3x_1+3x_2$$
$$-3x_1- x_2 \leq 25$$
$$2x_1+ x_2 \leq 5$$
$$-5x_1 + 2x_2 \leq 50$$

Il problema 2. non ammette soluzioni, dunque posso concludere che anche il problema 1. abbia come insieme delle soluzioni ammissibili l'insieme vuoto.

6.2 Come si comporta l'algoritmo in presenza di casi particolari?

Un caso molto interessante da affrontare è vedere fino a quanto può essere preciso il nostro metodo risolutivo, ovviamente dipenderà anche dalla precisione della funzione linprog usata da matlab. Nei capitoli precedenti, abbiamo evidenziato come grazie all'approccio preemptive si riesca a distinguere problemi (descritti da vincoli stretti) risolubili e non, ma cosa succede nel caso di due vincoli molto "vicini" tra loro.
Per chiarire le idee, riprendiamo un esempio trattato in precedenza (**P3**).

$$\min x_1+ x_2$$
$$-2x_1+ x_2 \leq 1$$
$$x_1- 2x_2 \leq -4$$
$$x_1+ x_2 \leq 14 - 10^{-3}$$
$$-x_1- x_2 \leq -14 - 10^{-3}$$

Le due rette che delimitano i due semipiani descritti dai vincoli 3 e 4, sono parallele tra loro. Intuitivamente per via grafica sappiamo che il problema così descritto non ammette soluzioni ammissibili.
Come prima cosa andiamo a vedere se il nostro algoritmo riesce a riportare lo stesso risultato, e avvicinando sempre più le due rette, anche con quale precisione ci riesce.

Dunque applicando l'algoritmo definitivo al problema:

1. Risolvo il corrispettivo del problema nella forma $Ax \leq b$
2. Linprog restituisce : "No feasible solution found"
3. Posso concludere subito che il problema iniziale $Ax < b$ non ha soluzioni

Passo dunque a risolvere il seguente problema:

$$\min x_1 + x_2$$
$$-2x_1 + x_2 \leq 1$$
$$x_1 - 2x_2 \leq -4$$
$$x_1 + x_2 \leq 14 - 10^{-8}$$
$$-x_1 - x_2 \leq -14 - 10^{-8}$$

Applicando nuovamente l'algoritmo:

1. Risolvo il corrispettivo del problema nella forma $Ax \leq b$
2. Linprog restituisce un valore ottimo x = [4.3333, 9.6667].

Ci troviamo nel caso di un problema apparentemente compatibile. Dunque proseguo con l'algoritmo utilizzando l'approccio preemptive.

3. Lo trasformo in forma DLEX e risolvo iterativamente i due step
4. Al primo step l'ottimo del duale è y = [0,0,3,0]
5. Al secondo step il problema duale è "unbounded"

Dalla teoria della dualità se il problema duale è unbounded, il corrispettivo primale è unfeasible, cioè non ammette soluzioni ammissibili.

Possiamo concludere che l'algoritmo appena descritto riesca a rilevare e risolvere, problemi apparentemente compatibili. Superando così i problemi causati dalle approssimazioni di linprog.

6.3 Esempio in R^{10}

In questo capitolo viene presentato un esempio meno ovvio di quelli già trattati. Il problema in questione è il seguente:

$$\text{LexMax } x_1, x_2, \ldots\ldots, x_9, x_{10}$$

$$\text{St. } \{ x \in R^{10} : -1 < x_i < 1, i = 1\ldots..10 \}$$

Il dominio descritto da questo problema è un ipercubo in R^{10} centrato nell'origine. Per rendere le cose ancora più interessanti, applichiamo una rotazione randomica al cubo. L'algoritmo usato per la rotazione è descritto in [8].
In questo caso la funzione obiettivo scelta non è tanto importante, anch'essa generata casualmente. Il problema così trasformato è il seguente:

$$\text{Max } c * x$$

$$\text{s.t. } \{\mathbf{x}' \in R^{10} : \mathbf{A}'\mathbf{x}' < \mathbf{b}'\}.$$

Dove la matrice **A'** e vettore **b'** sono le seguenti:

$$\mathbf{A}' = \begin{bmatrix}
-0.72, & -0.38, & 0.27, & 0.22, & 0.02, & 0.34, & -0.06, & -0.26, & -0.11, & 0.08 \\
0.12, & -0.31, & 0.15, & -0.34, & -0.42, & -0.35, & 0.08, & -0.38, & -0.49, & -0.24 \\
0.00, & 0.53, & -0.09, & -0.26, & 0.26, & 0.28, & -0.04, & -0.70, & -0.05, & -0.08 \\
-0.21, & -0.09, & -0.02, & -0.81, & 0.08, & 0.06, & -0.19, & 0.25, & 0.02, & 0.42 \\
-0.49, & 0.28, & 0.02, & -0.22, & 0.01, & -0.14, & 0.16, & 0.30, & 0.08, & -0.70 \\
-0.21, & 0.21, & 0.05, & 0.15, & 0.49, & -0.54, & 0.23, & 0.04, & -0.45, & 0.32 \\
-0.16, & 0.18, & 0.36, & 0.00, & -0.27, & -0.47, & 0.03, & -0.28, & 0.63, & 0.24 \\
0.20, & 0.29, & 0.74, & -0.04, & -0.12, & 0.35, & 0.32, & 0.22, & -0.19, & 0.09 \\
0.09, & 0.10, & 0.38, & 0.08, & 0.17, & -0.15, & -0.85, & 0.07, & -0.12, & -0.18 \\
-0.25, & 0.47, & -0.28, & 0.17, & -0.63, & 0.05, & -0.21, & 0.12, & -0.30, & 0.25 \\
0.72, & 0.38, & -0.27, & -0.22, & -0.02, & -0.34, & 0.06, & 0.26, & 0.11, & -0.08 \\
-0.12, & 0.31, & -0.15, & 0.34, & 0.42, & 0.35, & -0.08, & 0.38, & 0.49, & 0.24 \\
-0.00, & -0.53, & 0.09, & 0.26, & -0.26, & -0.28, & 0.04, & 0.70, & 0.05, & 0.08 \\
0.21, & 0.09, & 0.02, & 0.81, & -0.08, & -0.06, & 0.19, & -0.25, & -0.02, & -0.42 \\
0.49, & -0.28, & -0.02, & 0.22, & -0.01, & 0.14, & -0.16, & -0.30, & -0.08, & 0.70 \\
0.21, & -0.21, & -0.05, & -0.15, & -0.49, & 0.54, & -0.23, & -0.04, & 0.45, & -0.32 \\
0.16, & -0.18, & -0.36, & -0.00, & 0.27, & 0.47, & -0.03, & 0.28, & -0.63, & -0.24 \\
-0.20, & -0.29, & -0.74, & 0.04, & 0.12, & -0.35, & -0.32, & -0.22, & 0.19, & -0.09 \\
-0.09, & -0.10, & -0.38, & -0.08, & -0.17, & 0.15, & 0.85, & -0.07, & 0.12, & 0.18 \\
0.25, & -0.47, & 0.28, & -0.17, & 0.63, & -0.05, & 0.21, & -0.12, & 0.30, & -0.25
\end{bmatrix},$$

$$\mathbf{b}' = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T.$$

Questo problema ovviamente può essere risolto utilizzando l'algoritmo presentato precedentemente. Effettuando tutti i passaggi ottengo che il problema è feasible (ce lo aspettavamo, anche per la costruzione del problema stesso). L'ottimo del problema è:

X = [-2.4524, 0.051, -0.7682, 0.0679, -0.1854, -0.8211, 0.4559, 0.9504, 0.7091, 1.0194]

Per testare il caso particolare relativo a questo problema aggiungiamo un vincolo (ovviamente il nuovo vincolo è descritto da una disequazione lineare stretta). Avremo tre casi che andremo a trattare:

1) Vincolo "al di fuori" dell'ipercubo
2) Vincolo "dentro" l'ipercubo
3) Vincolo che comprende il bordo dell'ipercubo (ma fuori dall'ipercubo)

Primo caso

Per vincolo "al fuori" dell'ipercubo si intende un vincolo che descrive un semispazio che non si interseca con il dominio dell'ipercubo descritto dalla equazione $\mathbf{A'x'} < \mathbf{b'}$.
In questo caso, applicando l'algoritmo, allo step 0, quando risolvo il corrispondente problema $\mathbf{A'x'} \leq \mathbf{b'}$ con il simplesso, si ottiene come risultato l'insieme vuoto, e dunque posso già concludere che il problema iniziale
$\mathbf{A'x'} < \mathbf{b'}$ è un problema unfeasible, per le spiegazioni precedenti. (problema chiaramente incompatibile)

Secondo caso

Per vincolo "dentro" l'ipercubo si intende un vincolo che descrive un semispazio che interseca il dominio dell'ipercubo, formando così un sottospazio che sarà il nuovo dominio del mio problema.
La trattazione di questo caso è banale, questo problema si risolve normalmente utilizzando l'algoritmo definitivo, semplicemente applicato al nuovo dominio. In questo caso potrebbe restituire ovviamente un'altra soluzione ottima del problema.

Terzo caso

Nel caso in cui vincolo si trovasse sul bordo dell'ipercubo, la trattazione non è più così semplice. Utilizzando sempre lo stesso algoritmo definitivo, allo step 0 troverò un insieme di soluzioni ammissibili, costituiti dai punti che formano il bordo. A questo non posso ancora concludere niente perchè il problema è per il momento apparentemente compatibile, passo agli step successivi. Alla fine dello step 3, otterremo che il problema DLEX corrispondente è "Unbounded". Dunque posso concludere che il problema è incompatibile.

6.4 Conclusioni

Lo scopo di questa tesi era trovare un modo algoritmico per risolvere un problema apparentemente banale, cioè la risoluzione di un problema di fattibilità in forma $\mathbf{Ax} < \mathbf{b}$. Tale problema sembra banale perché, ad un occhio inesperto, sembra un problema molto simile ai più famosi problemi di fattibilità di programmazione lineare classici aventi dominio descritto mediante vincoli espressi nella forma $\mathbf{Ax} \leq \mathbf{b}$, che hanno molteplici algoritmi ottimizzati per la loro risoluzione.

Invece il problema preso in considerazione è di natura completamente differente, in quanto, nel nostro caso, il dominio è aperto invece che chiuso (la frontiera non fa parte del dominio). Questo è un aspetto che manda immediatamente in crisi l'algoritmo del simplesso tradizionale.

L'algoritmo definitivo proposto in questa tesi, pur basandosi sull'algoritmo del simplesso, riesce a risolvere il problema in maniera efficace ed esaustiva, e senza richiedere ulteriori parametri da settare da parte dell'utente.

RIFERIMENTI

[1] Pappalardo M. , Passacantando M. : *Ricerca Operativa (Pisa University Press, 2012) 15-83*

[2] Dantzig, G.B., Thapa, M.N.: Linear Programming 1: Introduction. Springer, New York, NY (1997). https://doi.org/10.2307/3010109

[3] Dantzig, G.B., Thapa, M.N.: Linear Programming 2: Theory and Extensions. Springer, New York, NY (2003). https://doi.org/10.1007/b97283

[4] Benci V., Di Nasso M.: *How to Measure the Infinite (World Scientific, 2019)*
pp 47-73. https://doi.org/10.1142/9789812836380_0003

[5] Cococcioni M., Fiaschi F.: *Linear Programming with Non-Archimedean Right-Hand Sides* (2023) https://doi.org/10.21203/rs.3.rs-2772714/v1

[6] Sergeyev, Y.D., De Leone, R.: Numerical Infinities and Infinitesimals in Optimization vol. 43, ECC. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-93642-6

[7] Sergeyev, Y.D.: Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems. EMS Surveys in Mathematical Sciences 4, 219–320 (2017). https://doi.org/10.4171/emss/ 4-2-3

[8] Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm. Applied Mathematics and Computation 318, 298–311 (2018). https://doi.org/10.1016/j.amc.2017.05.058

RINGRAZIAMENTI

Ho pensato a lungo cosa scrivere in questa pagina, non è banale per me farlo, devo tanto a tante persone che mi sono state vicine in questi tre anni e non solo. Nella mia vita sono cresciuto con la convinzione di dovercela fare sempre da solo e che meno aiuto chiedi e meglio è. Ma non è mai stato così, ho avuto la fortuna di esser stato circondato da persone che mi hanno sempre dato una mano. Per questo desidero ringraziarvi tutti.
In primis ci tengo a ringraziare il mio relatore, il professor Marco Cococcioni, per la disponibilità e il supporto che mi ha dato in questi mesi per la stesura della tesi.

Ai miei compagni di uni, senza di voi probabilmente non sarei nemmeno qui a laurearmi oggi, pochi sanno che durante il primo anno avrei voluto tanto mollare, ma poi ho conosciuto voi. Ad Au e Dario per avermi regalato persino la gioia di prendere il treno la mattina, alle mie amiche Chia, Ele, Vale, Giuli, Cate, Marti per tutte le serate, gli aperitivi e le risate, a Gionni, Tachi, Caio, Alex e Matte per tutte le chiacchierate, i consigli che mi avete dato, e anche tutti i ragazzi che ho conosciuto più recentemente, ognuno di voi mi ha lasciato qualcosa. Grazie davvero, conserverò con cura tutti i ricordi di questi ultimi anni, grazie per ogni volta che mi avete ascoltato, aiutato o semplicemente abbiamo pranzato insieme.

Ai miei bimbi di Livorno, vi voglio tanto bene, grazie perchè siete sempre stati come una famiglia. Non importava quando o come, o se c'erano momenti in cui non ci sentivamo spesso, sapevo in modo o nell'altro che voi eravate lì. Grazie ai bimbi che sono venuti a dare una mano a me e alla mia famiglia al ristorante quando avevamo bisogno (o nelle sere in cui facevo serate a Pisa :D). Grazie per esserci stati, sappiate che ora a Milano avete un amico in più su cui contare.

Ai miei amici di sempre, a Nicola per essere sempre stato al mio fianco sia nei banchi di scuola che nella mia vita, a Gio che anche se non ci sentiamo spesso ci vogliamo tanto bene, a Ila che continua a mangiare gli sbelli, a Matte che ci conosciamo da una vita e non ti ho mai detto che a modo tuo sei una mia fonte di ammirazione.

Alla mia famiglia, grazie 妈妈 e 爸爸 per tutto quello che mi avete dato, grazie per tutti i sacrifici che avete fatto per non farmi mai mancare nulla. Siete i miei eroi.

Infine a Marta, la mia ancora di salvezza e la mia casa sicura in cui rifugiarmi. Siamo una squadra, grazie per supportarmi e sopportarmi in tutto ciò che faccio. Grazie per essere la persona meravigliosa che sei.

Una dedica speciale a 奶奶 e Diego.