УНИВЕРЗИТЕТ “Св. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ

ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

- ДИПЛОМСКА РАБОТА -

по предметот

ОСНОВИ НА КОНВЕКСНА ОПТИМИЗАЦИЈА СО ПРИМЕНА

Тема

СЕМИДЕФИНИТНА РЕЛАКСАЦИЈА НА КВАДРАТНИ ОПТИМИЗАЦИСКИ ПРОБЛЕМИ

Ментор:
Проф. д-р Катерина Хаџи-Велкова Санева

Изработил:
Надежда Илиева, индекс бр. 17/2018
e-mail: ilieva.nadezda@gmail.com

Скопје, јуни 2022

Содржина

Листа на слики

Апстракт

Семидефинитната релаксација е моќна апроксимациска техника која се применува во решавањето на класа неконвексни квадратни програми со квадратни ограничувања. Во оваа дипломска работа даваме краток вовед во квадратните програми со квадратни ограничувања и семидефинитното програмирање, за потоа да го воведеме и анализираме концептот на семидефинитна релаксација. Ги изложуваме методите за извлекување на погодно решение на оригиналниот проблем од оптималното решение на неговата релаксирана форма – методот на апроксимација со сопствени вектори и методот на рандомизација. Ја разгледуваме поширокатa примена на семидефинитната релаксација, за да на крај целосно прикажеме нејзина практична имплементација во проблемот на оптимална детекција на пренесените симболи во MIMO системите. MIMO ML проблемот најпрвин го моделираме како бинарна квадратна програма, која потоа ја релаксираме во конвексен проблем за чие практично решавање го користиме софтверскиот пакет CVX во MATLAB. За крај вршиме споредба помеѓу SDR детекторите со апроксимација со сопствени вектори (ранк-1 апроксимација) и рандомизација и детекторот претставник од класата на линеарни детектори. Со симулации ги потврдуваме добрите перформанси на SDR детекторите од аспект на помала веројатност за битска грешка и полиномна комплексност.

Клучни зборови: Квадратна програма со квадратни ограничувања; Семидефинитно програмирање; Семидефинитна релаксација; MIMO; ML детектор.

Вовед

Од распоредувањето на атомите во кристални решетки до правилните шестаголни клетки во пчелиното саќе, природата изобилува со примери на оптимални системи. Како во природата, така и во инженерството постои стремеж за изнаоѓање на дизајни или постапки кои ќе бидат најдобри, најбрзи, најефективни и најекономични. За постигнување на овие цели се користат техники и алгоритми од областа на математичката оптимизација, позната под името математичко програмирање.

Процесот на изнаоѓање на оптимално решение на конкретен инженерски проблем понекогаш е далеку од лесен и ефикасен. Во пракса секогаш постои компромис помеѓу точноста на апроксимираното решение и комплексноста на алгоритмот за одредување на истото. Дел од проблемите кои се јавуваат во реалноста се моделираат како NP-тешки (анг. nondeterministic polynomial-time hardness) неконвексни проблеми, за чие точно решавање досега познатите алгоритми достигнуваат експоненцијална комплексност. Од тие причини, голем е интересот за развивање и примена на техники кои нудат ефикасно приближно решавање на ваквите проблеми.

Една таква моќна и пресметковно ефикасна апроксимациска техника која во најлош случај има полиномна комплексност е семидефинитната релаксација, [1], [2], [5], [9], [13]. Семидефинитната релаксација наоѓа примена во решавањето на класа неконвексни проблеми од типот на квадратни програми со квадратни ограничувања и истата ќе биде предмет на оваа дипломска работа. Притоа ќе биде илустрирана интуицијата зад оваа метода, нејзината имплементација, како и практичната примена.

Класата на неконвексни квадратни програми со квадратни ограничувања вклучува проблеми кои се од голем интерес во телекомуникациите и процесирањето на сигнали. Пример за таков проблем е бинарната квадратна програма, која е NP-тежок пресметковен проблем. Можноста да се реши бинарната квадратна програма има големо значење во детекцијата на пренесените симболи во MIMO (multiple-input multiple-output) системите, кои се појавуваат во многу модерни телекомуникациски канали. Практичната имплементација на детектор со семидефинитна релаксација ќе биде предмет на анализа во оваа дипломска работа. Преку симулации ќе бидат разгледани и потврдени перформансите и комплексноста на овој квази-оптимален детектор, кој се карактеризира како ефикасен пристап со високи перформанси, [1], [11], [12].

1 Квадратно програмирање со квадратни ограничувања

Ќе започнеме со дефинирање на основната терминологија која ќе се користи во оваа дипломска работа. Нотацијата

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \ & f_0(\boldsymbol{x}) \\ s.t. \ & f_i(\boldsymbol{x}) \le 0, \quad i=1,...,m, \\ & h_i(\boldsymbol{x}) = 0, \quad i=1,...,p \end{aligned} \tag{1.1}$$

ќе ја користиме за да го опишеме оптимизацискиот проблем на наоѓање на вредноста на променливата $\boldsymbol{x}$ која ја минимизира функцијата $f_0(\boldsymbol{x})$ и ги задоволува условите (ограничувањата): $f_i(\boldsymbol{x}) \le 0,\ i=1,...,m$ и $h_i(\boldsymbol{x}) = 0,\ i=1,...,p$. Променливата $\boldsymbol{x} \in \mathbb{R}^n$ се нарекува *оптимизациска променлива*, а функцијата $f_0 : \mathbb{R}^n \to \mathbb{R}$ претставува *целна функција*. Функциите $f_i : \mathbb{R}^n \to \mathbb{R},\ i=1,...,m$ и $h_i : \mathbb{R}^n \to \mathbb{R},\ i=1,...,p$ се *ограничувачки функции*. Во случај кога (1.1) нема ограничувања станува збор за *безусловна оптимизација*. Секаде понатаму во дефинирањето на оптимизациските проблеми ја користиме кратенката s.t. од анг. “subject to” („под услов“).

Конвексен оптимизациски проблем е проблем од облик:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \ & f_0(\boldsymbol{x}) \\ s.t. \ & f_i(\boldsymbol{x}) \le 0, \quad i=1,...,m, \\ & \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i=1,...,p, \end{aligned} \tag{1.2}$$

каде што $\boldsymbol{a}_1,...,\boldsymbol{a}_p \in \mathbb{R}^n$, $b_1,...,b_p \in \mathbb{R}$ и функциите $f_0,...,f_m$ се конвексни функции[1]. За разлика од проблемот (1.1), кај конвексниот проблем целната функција и ограничувачките функции со знак на неравенство мораат да бидат конвексни, додека ограничувачките функции $h_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} - b_i = 0$ мораат да бидат афини[2].

Конвексниот оптимизациски проблем (1.2) се нарекува *квадратна програма* ако целната функција е (конвексна) квадратна, а ограничувачките функции се афини. Квадратната програма може да се изрази како

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \ & \frac{1}{2}\boldsymbol{x}^T \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} + r \\ s.t. \ & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{1.3}$$

---

[1] Функцијата $f : \mathbb{R}^n \to \mathbb{R}$ е *конвексна функција* ако нејзиниот домен е конвексно множество и за $\forall \boldsymbol{x}, \boldsymbol{y}$ во нејзиниот домен и за $\forall \lambda \in [0,1]$ е исполнето

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \le \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

[2] Функцијата $f : \mathbb{R}^n \to \mathbb{R}^m$ е *афина функција* ако е збир на линеарна функција и константа, односно ако ја има формата

$$f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b},$$

каде $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ и $\boldsymbol{b} \in \mathbb{R}^m$.

каде што $\boldsymbol{P}$ е реална $n \times n$ симетрична матрица (означуваме $\boldsymbol{P} \in \mathbb{S}^n$, каде $\mathbb{S}^n$ го означува множеството на реални симетрични $n \times n$ матрици), позитивно семидефинитна матрица (означуваме $\boldsymbol{P} \succeq 0$, односно $\boldsymbol{z}^T \boldsymbol{P} \boldsymbol{z} \geq 0$ за $\forall \boldsymbol{z} \in \mathbb{R}^n$), $\boldsymbol{G} \in \mathbb{R}^{m \times n}$, $\boldsymbol{A} \in \mathbb{R}^{p \times n}$, $\boldsymbol{q} \in \mathbb{R}^n$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{b} \in \mathbb{R}^p$ и $r \in \mathbb{R}$. Притоа, ознаката '$\preceq$' во (1.3) означува помало или еднакво по координати, односно $G_{i1} x_1 + G_{i2} x_2 + ... + G_{in} x_n \leq h_i$ за $i = 1, ..., m$. Ако целната функција и ограничувањата во облик на неравенство во (1.3) се (конвексни) квадратни функции, како во

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \ & \frac{1}{2} \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x} + \boldsymbol{q}_0{}^T \boldsymbol{x} + r_0 \\ s.t. \ & \frac{1}{2} \boldsymbol{x}^T \boldsymbol{P}_i \boldsymbol{x} + \boldsymbol{q}_i{}^T \boldsymbol{x} + r_i \leq 0, \quad i = 1, ..., m, \\ & \boldsymbol{A} \boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{1.4}$$

каде што $\boldsymbol{P}_i \in \mathbb{S}^n$, $\boldsymbol{P}_i \succeq 0$, $\boldsymbol{q}_i \in \mathbb{R}^n$ и $r_i \in \mathbb{R}$ за $i = 0, 1, ..., m$, тогаш проблемот се нарекува *квадратна програма со квадратни ограничувања*. Ако е исполнето дека $\boldsymbol{P}_i \succeq 0$ за $i = 0, 1, ..., m$, како во проблемот (1.4), тогаш проблемот е конвексен и ефикасно решлив, [2]. Меѓутоа, ова не мора да биде случај за сите квадратни програми со квадратни ограничувања. Имено, квадратната програма со квадратни ограничувања е оптимизациски проблем кој општо може да се запише како

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \ & \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x} + \boldsymbol{q}_0{}^T \boldsymbol{x} + r_0 \\ s.t. \ & \boldsymbol{x}^T \boldsymbol{P}_i \boldsymbol{x} + \boldsymbol{q}_i{}^T \boldsymbol{x} + r_i \leq 0, \quad i = 1, ..., m, \end{aligned} \tag{1.5}$$

со таа разлика што во овој случај за матриците $\boldsymbol{P}_i \in \mathbb{S}^n$ не се претпоставуваат никакви дополнителни услови за нивната дефинитност. Генерално, тоа значи дека проблемот (1.5) е *неконвексен оптимизациски проблем*.

Постојат алтернативни еквивалентни формулации на проблемот (1.5). Една таква формулација е *епиграфската форма*[3], која, без губење на општоста, целната функција ја прави линеарна (конвексна) преку воведување на дополнителна променлива $t \in \mathbb{R}$:

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n, t \in \mathbb{R}} \ & t \\ s.t. \ & \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x} + \boldsymbol{q}_0{}^T \boldsymbol{x} + r_0 \leq t \\ & \boldsymbol{x}^T \boldsymbol{P}_i \boldsymbol{x} + \boldsymbol{q}_i{}^T \boldsymbol{x} + r_i \leq 0, \quad i = 1, ..., m. \end{aligned} \tag{1.6}$$

Исто така, возможно е да се запише проблем еквивалентен на проблемот (1.5) кој има само квадратни ограничувања со знак на еднаквост, преку воведување на дополнителна промелнива $\boldsymbol{s} \in \mathbb{R}^m$:

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \ & \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x} + \boldsymbol{q}_0{}^T \boldsymbol{x} + r_0 \\ s.t. \ & \boldsymbol{x}^T \boldsymbol{P}_i \boldsymbol{x} + \boldsymbol{q}_i{}^T \boldsymbol{x} + r_i + s_i^2 = 0, \quad i = 1, ..., m. \end{aligned} \tag{1.7}$$

---

[3] *Епиграф* или *суперграф* на функцијата $f : X \to \mathbb{R}$ е множеството на сите точки од Декартовиот производ $X \times \mathbb{R}$ кои лежат над или на нејзиниот график, односно

$$\operatorname{epi} f = \{(x, r) \in X \times \mathbb{R} \mid r \geq f(x)\}.$$

Неконвексните квадратни програми со квадратни ограничувања се вбројуваат во класата на т.н. NP-тешки (анг. nondeterministic polynomial-time hardness) проблеми. Не е докажано дека овие проблеми се навистина тешки, но практично сите познати алгоритми кои можат да се искористат во нивното решавање имаат комплексност која расте експоненцијално со големината на проблемот, [2]. Пред да изложиме неколку популарни практични примери на неконвексни квадратни програми со квадратни ограничувања, ќе дадеме кратко објаснување за начинот на дефинирање и утврдување на комплексноста на некој алгоритам, како и за значењето на поимот NP-тежина.

1.1 Алгоритамска комплексност

Алгоритам е добро осмислена постапка од конечен број чекори за решавање на некој проблем. Решението на пресметковен проблем се смета за *ефикасно* доколку се вклопува во расположливите граници на употребените ресурси (расположливата меморија и прифатливото време на извршување). За да се измери ефикасноста на даден алгоритам потребно е да се одреди неговата пресметковна комплексност, која претставува одраз на неговото трошење на клучниот ресурс – времето на извршување. Со други зборови, алгоритамската комплексност покажува колку брзо се извршува алгоритамот и се дефинира како нумеричка функција $f(n)$, каде n ја претставува големината на влезниот параметар. Експерименталното одредување на времето на извршување е прецизно и лесно изводливо, но таквото мерење не дава објективна ниту компаративна слика за алгоритамот бидејќи бројот n не е единствениот параметар. На времето на извршување дополнително влијаат и хадрверските (CPU, RAM, HDD) и софтверските (оперативен систем, развоен јазик, други таскови) специфичности на околината во која се прави експериментот. Поради овие причини како генерална алатка за утврдување на комплексноста $f(n)$ се користи асимптотската анализа. Суштински чекор во асимптотската анализа е одредување на аналитичката форма на $f(n)$ како функција од кардиналноста на влезот n и утврдување на нејзината асимптотска граница (при $n \to \infty$) изразена преку друга аналитички позната математичка функција.

При определување на временската комплексност $f(n)$ од интерес е времето на извршување изразено како збир на елементарни (примитивни) операции. Во овие операции спаѓаат: доделување на вредност на променлива, повикување на функција, извршување на аритметичка операција (собирање, одземање, множење, делење), споредба на две променливи, враќање на резултат од функција и други. Аналитичкото одредување на комплексноста ќе го илустрираме преку примерот прикажан на сликата 1.1.1.

```
int sum = 0;                    // доделување -> 1
for (int i = 1; i<=n; i++) {    // 1 доделување, n + 1 споредба и n инкрементирања -> 2*n + 2
    sum += n;                   // n собирања и n доделувања -> 2 * n
}
```

Слика 1.1.1 Пример за аналитичко одредување на комплексноста

Бидејќи имаме едно доделување на вредност на променлива пред for јамката (`int sum = 0`), проследено со едно доделување (`int i = 1`), $n+1$ споредба (`i<=n`), n инкрементирања (`i++`), n собирања и n доделувања (`sum += n`) во рамки на истата, аналитичката форма на $f(n)$ може да се изрази како:

$$f(n) = 1+1+(n+1)+n+2n = 4n+3. \qquad \textbf{(1.1.1)}$$

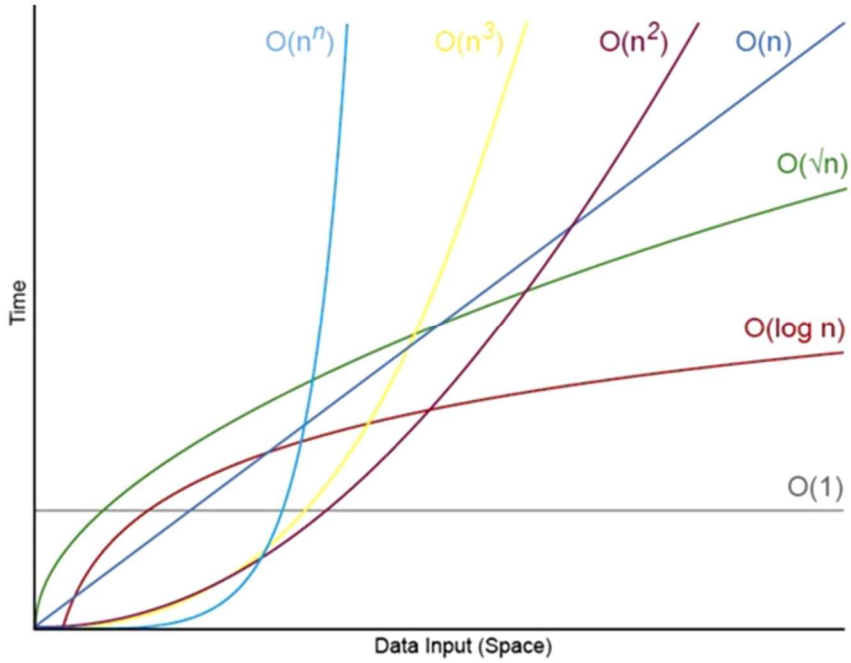
Како што споменавме и погоре, со цел да се избегне прецизна евалуација на комплицирани функции, асимптотската анализа предлага проценка на комплексноста $f(n)$ преку споредба со друга позната функција $g(n)$. Може да се каже дека асимптотската анализа е апроксимативна, односно дека ги елиминира помалку значајните кардиналности на n во аналитичката форма на $f(n)$ со цел фокус на компонентата од изразот која го дава најзначајниот придонес во брзината на раст. Една од асимптотските нотации кои се користат за опишување на комплексноста на алгоритмите е O (големо О), која ја одразува горната граница, т.е. максималниот број на чекори за решавање на проблемот. O нотацијата одговара на растот на најлошата варијанта и нејзината формална дефиниција е дадена во продолжение:

Нека се дадени се функциите $f:\mathbb{N}\to\mathbb{R}$ и $g:\mathbb{N}\to\mathbb{R}$. Велиме дека $f(n)$ е $O(g(n))$ ако постојат реална константа $c>0$ и целобројна константа $n_0>0$ такви што

$$f(n)\le c\cdot g(n), \qquad \textbf{(1.2.1)}$$

за $n\ge n_0$. Константата n_0 ја претставува најмалата големина на влезот за која е исполнето неравенството (1.2.1). Дефиницијата тврди дека и во најлошиот случај за n, кое е доволно големо и над n_0, алгоритамот ќе се извршува во траење (број на чекори) од најмногу $c\cdot g(n)$.

Се разликуваат константни функции $O(1)$, логаритам-логаритам функции $O(\lg\lg n)$, логаритамски функции $O(\lg n)$, полилогаритамски функции $O(\lg^k n)$, коренски функции $O(n^{\frac{1}{k}})$, линеарни фунцкии $O(n)$, надлинеарни функции $O(n\lg n)$, квадратни функции $O(n^2)$, полиномни функции $O(n^k)$, $k>1$, и експоненцијални функции $O(2^n);O(n^n);O(n!)$, ..., [3]. На сликата 1.1.2 е направена споредба на неколку од споменатите алгоритамски комплексности опишани преку O нотацијата.



Слика 1.1.2 Споредба на различни алгоритамски комплексности

Генерално алгоритмите се класифицираат во две поширoки категории: *полиномни алгоритми* и *експоненцијални алгоритми*. Полиномните алгоритми се алгоритми чија комплексност е пропорционална или ограничена со некоја полиномна функција од големината на влезот n. За разлика од полиномните, експоненцијалните алгоритми се алгоритми кои за поголеми вредности на влезниот параметар n не се придржуваат кон полиномна граница. Полиномните алгоритми се сметаат за *добри*, односно *ефикасни*, додека експоненцијалните алгоритми се сметаат за *лоши*, односно *неефикасни*, [14].

За крај, класата на т.н. NP (анг. nondeterministic polynomial-time) проблеми ги опфаќа проблемите што можат да се решат во полиномно време со недетреминистичка Тјурингова машина. Дополнително, во пресметковната теорија на комплексност со NP-тешки проблеми (анг. NP-hardness, nondeterministic polynomial-time hardness) се опишуваат проблемите кои неформално се барем толку тешки за решавање колку најтешките NP проблеми.

1.2 Бинарни квадратни програми

Бинарната квадратна програма ја има следната форма

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} \\ &s.t. \quad x_i^2 = 1, \quad i = 1,...,n, \end{aligned} \tag{1.2.1}$$

каде матрицата $\boldsymbol{C} \in \mathbb{S}^n$. Оваа програма претставува пресметковно тежок проблем, кој спаѓа во класата на NP-тешки проблеми, [1].

1.3 Бинарни проблеми на најмали квадрати

Бинарен проблем од семејството на најмали квадрати е

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 \\ &s.t. \quad x_i \in \{-1,1\}, \quad i = 1,...,n, \end{aligned} \tag{1.3.1}$$

каде $\boldsymbol{A} \in \mathbb{R}^{k\times n}\,(k \geq n)$, $\boldsymbol{x} \in \{\pm 1\}^n$ и $\boldsymbol{b} \in \mathbb{R}^k$. Ваквиот проблем наоѓа примена во дигиталните телекомуникации во процесот на ML (Maximum Likelihood) детекција на дигиталните сигнали. Стратегија со „груба сила“ (анг. brute force) за решавање на проблемот (1.3.1) е проверка на сите 2^n можни вредности за векторот $\boldsymbol{x}$. Проблемот (1.3.1) може да се изрази како неконвексна квадратна програма со квадратни ограничувања на следниот начин:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b} \\ &s.t. \quad x_i^2 = 1, \quad i = 1,...,n\,. \end{aligned} \tag{1.3.2}$$

1.4 MAXCUT проблем

Нека е даден ненасочен тежински граф G со множество на темиња $V = \{1,2,...,n\}$, гранки (ребра) $E \subset \{\{i,j\} \mid i,j \in V, i \neq j\}$ и тежини w_{ij} на секоја гранка $\{i,j\}$, каде $w_{ij} = 0$ за $\{i,j\} \notin E$. MAXCUT проблемот бара *пресек* на графот G со најголема тежина, односно поделба на множеството на темиња на две партиции V_1 и

V_2 ($V = V_1 \sqcup V_2$), такви што збирот на тежините на гранките кои ги поврзуваат овие две партиции да биде максимален. Со други зборови, се максимизира сумата

$$cut(V_1, V_2) \coloneqq \sum_{i \in V_1,\, j \in V_2} w_{ij} \tag{1.4.1}$$

над сите можни партиции V_1 и V_2 на множеството V, односно, MAXCUT проблемот може да се формулира како

$$\max_{V_1, V_2} \; cut(V_1, V_2). \tag{1.4.2}$$

Лапласовата матрица на тежински граф $L(G,w)$ е $n \times n$ симетрична матрица која се дефинира на следниот начин:

$$L(G,w)_{ij} \coloneqq \begin{cases} -w_{ij}, & ако \{i,j\} \in E \\ \sum_k w_{ik}, & ако\ i = j \\ 0, & во\ друг\ случај. \end{cases} \tag{1.4.3}$$

За даден пресек (V_1, V_2) на V, на сите темиња во V_1 им се придружува 1, додека на сите темиња во V_2 им се придружува -1. На овој начин, пресекот на графот G може да се разгледува преку вектор $\boldsymbol{x} \in \mathbb{R}^n$, таков што $x_i \in \{1, -1\}$, односно $x_i^2 = 1$ за $i = 1, ..., n$. Односно, за секое теме i во V ќе важи дека

$$x_i = \begin{cases} 1, & за\ i \in V_1 \\ -1, & за\ i \in V_2. \end{cases} \tag{1.4.4}$$

Според тоа, за величината $1 - x_i x_j$ за секоја гранка $\{i, j\} \in E$ ќе биде исполнето

$$1 - x_{ij} = \begin{cases} 0, & ако\ i\ и\ j\ се\ во\ истата\ партиција\ на\ V \\ 2, & ако\ i\ и\ j\ се\ во\ различна\ партиција\ на\ V. \end{cases} \tag{1.4.5}$$

Целната функција (1.4.1) може да се запише како

$$cut(V_1, V_2) = \sum_{i \in V_1,\, j \in V_2} w_{ij} = \frac{1}{2} \sum_{\{i,j\} \in E} w_{ij}(1 - x_i x_j) = \frac{1}{4} \sum_{i,j} L(G,w)_{ij} x_i x_j. \tag{1.4.6}$$

Според тоа, MAXCUT проблемот може да се преформулира во

$$\begin{aligned} &\max_{\boldsymbol{x} \in \mathbb{R}^n} \; \frac{1}{4} \boldsymbol{x}^T \boldsymbol{L(G,w)x} \\ &s.t. \quad x_i^2 = 1, \quad i = 1, ..., n, \end{aligned} \tag{1.4.7}$$

што претставува специјален случај на бинарната квадратна програма (1.2.1). MAXCUT проблемот е клучен проблем во теоријата на компјутерски науки и дополнително наоѓа примена во оптимизацијата на мрежи, физиката и дизајнот на електрични кола, [4].

2 Семидефинитно програмирање

Кај семидефинитното програмирање се минимизира линеарна функција подложена на позитивно семидефинитно ограничување кое е афина комбинација на симетрични матрици, односно

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{c}^T\boldsymbol{x} \\ &s.t. \quad F(\boldsymbol{x}) \succeq 0, \end{aligned} \tag{2.1}$$

каде што

$$F(\boldsymbol{x}) := \boldsymbol{F_0} + \sum_{i=1}^{n} x_i \boldsymbol{F}_i . \tag{2.2}$$

Векторот $\boldsymbol{c} \in \mathbb{R}^n$, а матриците $\boldsymbol{F}_0, \boldsymbol{F}_1, ..., \boldsymbol{F}_n \in \mathbb{S}^n$. Ограничувањето во проблемот (2.1) не е ниту линеарно ниту глатко, меѓутоа е конвексно, па семидефинитните програми се конвексни оптимизациски проблеми. Семидефинитните програми можат да се сретнат и во следната стандардна форма:

$$\begin{aligned} &\min_{\boldsymbol{X}\in\mathbb{S}^n} \ Tr(\boldsymbol{CX}) \\ &s.t. \quad Tr(\boldsymbol{A}_i\boldsymbol{X}) = b_i, \quad i = 1, ..., m \\ &\qquad \boldsymbol{X} \succeq 0, \end{aligned} \tag{2.3}$$

каде што

$$Tr(\boldsymbol{CX}) = \sum_{k=1}^{n}\sum_{j=1}^{n} C_{kj} X_{kj} \ , \tag{2.4}$$

$$Tr(\boldsymbol{A}_i\boldsymbol{X}) = \sum_{k=1}^{n}\sum_{j=1}^{n} A_{i,kj} X_{kj} \ , \tag{2.5}$$

и матриците $\boldsymbol{C}, \boldsymbol{A}_1, ..., \boldsymbol{A}_m \in \mathbb{S}^n$, [5], [13]. Притоа, Tr е ознака за траг на квадратна матрица, кој претставува збир на елементите на главната дијагонала.

Семидефинитното програмирање спојува повеќе стандардни проблеми, како линеарното и квадратното програмирање и наоѓа широка примена во инженерската и комбинаторната оптимизација. Иако семидефинитните програми се многу погенерални од линеарните програми, тие не се многу потешки за решавање. Многу од алгоритмите со внатрешна точка (анг. interior-point algorithms) за линеарното програмирање се генерализирани и се применуваат во семидефинитното програмирање со полиномна комплексност во најлош случај, [5].

3 Семидефинитна релаксација

Семидефинитната релаксација претставува моќна, пресметковно ефикасна апроксимациска техника која се применува во решавањето на многу тешки оптимизациски проблеми. Оваа метода може да се искористи во решавањето на неконвексни квадратни програми со квадратни ограничувања, во кои припаѓаат проблемите од видот

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} \\ &s.t. \quad \boldsymbol{x}^T\boldsymbol{F}_i\boldsymbol{x} \geq g_i, \quad i = 1, ..., p, \\ &\qquad \boldsymbol{x}^T\boldsymbol{H}_i\boldsymbol{x} = l_i, \quad i = 1, ..., q, \end{aligned} \tag{3.1}$$

каде матриците $\boldsymbol{C}, \boldsymbol{F}_1, ..., \boldsymbol{F}_p, \boldsymbol{H}_1, ..., \boldsymbol{H}_q$ се реални симетрични матрици кои можат да бидат и недефинитни, а $g_1, ..., g_p, l_1, ..., l_q \in \mathbb{R}$. За да биде нотацијата пократка, проблемот (3.1) може да се запише како

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} \\ &s.t. \quad \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x} \trianglerighteq_i b_i, \quad i = 1, ..., m, \end{aligned} \tag{3.2}$$

каде ознаката '$\trianglerighteq_i$' може да претставува '$\geq$', '$=$' или '$\leq$' за секое i. Матриците $\boldsymbol{C}, \boldsymbol{A}_1, ..., \boldsymbol{A}_m \in \mathbb{S}^n$, а $b_1, ..., b_m \in \mathbb{R}$. Важен прв чекор во изведувањето на семидефинитната релаксација на проблемот (3.2) е воочувањето на следното својство:

$$\begin{aligned} \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} &= Tr(\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}) = Tr(\boldsymbol{C}\boldsymbol{x}\boldsymbol{x}^T), \\ \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x} &= Tr(\boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x}) = Tr(\boldsymbol{A}_i\boldsymbol{x}\boldsymbol{x}^T), \end{aligned} \tag{3.3}$$

Во продолжение ќе го изложиме доказот на ова својство.

Доказ. Нека $\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}$ ја претставува квадратната форма на матрицата $\boldsymbol{C}$. Тогаш важи

$$\begin{aligned} \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} = {} & c_{11}x_1^2 + c_{12}x_1x_2 + ... + c_{1n}x_1x_n + \\ & + c_{21}x_2x_1 + c_{22}x_2^2 + ... + c_{2n}x_2x_n + ... + \\ & + c_{n1}x_nx_1 + c_{n2}x_nx_2 + ... + c_{nn}x_n^2. \end{aligned} \tag{3.4}$$

Матрицата $\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}$ има димензија 1×1, па ќе биде исполнето дека $Tr(\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}$. Од друга страна, лесно се пресметува дека

$$\begin{aligned} \boldsymbol{C}\boldsymbol{x}\boldsymbol{x}^T &= \begin{bmatrix} c_{11} & c_{12} & . & . & . & c_{1n} \\ c_{21} & c_{22} & . & . & . & c_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ c_{n1} & c_{n2} & . & . & . & c_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \begin{bmatrix} x_1 & x_2 & . & . & . & x_n \end{bmatrix} = \\ &= \begin{bmatrix} c_{11}x_1^2 + ... + c_{1n}x_nx_1 & c_{11}x_1x_2 + ... + c_{1n}x_2x_n & . & . & . & c_{11}x_1x_n + ... + c_{1n}x_n^2 \\ c_{21}x_1^2 + ... + c_{2n}x_nx_1 & c_{21}x_1x_2 + ... + c_{2n}x_2x_n & . & . & . & c_{21}x_1x_n + ... + c_{2n}x_n^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ c_{n1}x_1^2 + ... + c_{nn}x_nx_1 & c_{n1}x_1x_2 + ... + c_{nn}x_2x & . & . & . & c_{n1}x_1x_n + ... + c_{nn}x_n^2 \end{bmatrix}, \end{aligned} \tag{3.5}$$

од каде следува дека $Tr(\boldsymbol{C}\boldsymbol{x}\boldsymbol{x}^T) = \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}$. ■

Со користење на својството (3.3) се добива следната формулација за реалната квадратна програма со квадратни ограничувања (3.2):

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ Tr(\boldsymbol{C}\boldsymbol{x}\boldsymbol{x}^T) \\ &s.t. \quad Tr(\boldsymbol{A}_i\boldsymbol{x}\boldsymbol{x}^T) \trianglerighteq_i b_i, \quad i = 1, ..., m. \end{aligned} \tag{3.6}$$

Може да се забележи дека целната функција и ограничувањата се линеарни функции по матрицата $\boldsymbol{x}\boldsymbol{x}^T$. Дополнително, за секој вектор $\boldsymbol{x} \in \mathbb{R}^n$ важи дека матрицата $\boldsymbol{x}\boldsymbol{x}^T$ е:

а) симетрична матрица;
б) позитивно семидефинитна матрица;
в) матрица со ранг 1.

Доказ. а) Да забележиме дека

$$(\boldsymbol{x}\boldsymbol{x}^T)^T = (\boldsymbol{x}^T)^T \boldsymbol{x}^T = \boldsymbol{x}\boldsymbol{x}^T . \qquad (\mathbf{3.7})$$

со што покажуваме дека матрицата $\boldsymbol{x}\boldsymbol{x}^T$ е симетрична матрица.

б) Нека $\boldsymbol{y} \in \mathbb{R}^n$ е произволен реален вектор. Тогаш важи

$$\boldsymbol{y}^T(\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{y} = (\boldsymbol{x}^T\boldsymbol{y})^T(\boldsymbol{x}^T\boldsymbol{y}) = \left\|\boldsymbol{x}^T\boldsymbol{y}\right\|_2^2 \geq 0. \qquad (\mathbf{3.8})$$

Притоа, $\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^T\boldsymbol{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}$ е Евклидската норма на векторот $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$.

Според тоа, покажавме дека $\boldsymbol{x}\boldsymbol{x}^T$ е позитивно семидефинитна матрица.

в) За рангот на произволна матрица $\boldsymbol{A} \in \mathbb{R}^{k\times l}$ е исполнето

$$rank(\boldsymbol{A}) \leq \min\{k, l\}. \qquad (\mathbf{3.9})$$

Според тоа, ќе биде точно дека

$$rank(\boldsymbol{x}) \leq \min\{n, 1\} = 1, \qquad (\mathbf{3.10})$$

од каде следува

$$rank(\boldsymbol{x}) = 1 . \qquad (\mathbf{3.11})$$

Дополнително, за произволни матрици $\boldsymbol{A} \in \mathbb{R}^{k\times l}$ и $\boldsymbol{B} \in \mathbb{R}^{l\times m}$ важи

$$rank(\boldsymbol{A}\boldsymbol{B}) \leq rank(\boldsymbol{A}). \qquad (\mathbf{3.12})$$

Во нашиот случај

$$rank(\boldsymbol{x}\boldsymbol{x}^T) \leq rank(\boldsymbol{x}) = 1. \qquad (\mathbf{3.13})$$

со што добивамедека $rank(\boldsymbol{x}\boldsymbol{x}^T) = 1$. ■

Со воведување на нова променлива $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$, која според погоре покажаните особини, е симетрична позитивно семидефинитна матрица со ранг еднаков на 1, се добива следната формулација на проблемот (3.6), односно (3.2):

$$\begin{aligned} &\min_{\boldsymbol{X}\in\mathbb{S}^n} \; Tr(\boldsymbol{C}\boldsymbol{X}) \\ &s.t. \quad Tr(\boldsymbol{A}_i\boldsymbol{X}) \trianglerighteq_i b_i, \quad i = 1,\dots,m \\ &\qquad \boldsymbol{X} \succeq 0, \qquad rank(\boldsymbol{X}) = 1. \end{aligned} \qquad (\mathbf{3.14})$$

До овде изгледа како да се нема постигнато многу, бидејќи проблемот (3.14) е исто толку тежок за решавање како и проблемот (3.2). За разлика од формулацијата на проблемот (3.2), во формулацијата на проблемот (3.14) јасно може да се издвои ограничувањето $rank(\boldsymbol{X}) = 1$, кое го прави решавањето на овој проблем тешко. Ова ограничување не претставува конвексно ограничување. За споредба, целната функција и останатите ограничувања се конвексни функции по променливата $\boldsymbol{X}$. Главната идеја во овој момент е да се отфрли ограничувањето со рангот и да се добие следната формулација на проблемот (3.2):

$$\begin{aligned} &\min_{X\in\mathbb{S}^n} \; Tr(\boldsymbol{CX}) \\ &s.t. \quad Tr(\boldsymbol{A}_i\boldsymbol{X}) \trianglerighteq_i b_i, \quad i=1,...,m \\ &\qquad \boldsymbol{X} \succeq 0. \end{aligned} \tag{3.15}$$

Проблемот (3.15) претставува *семидефинитна релаксација* на проблемот (3.2). Називот 'семидефинитна' доаѓа од фактот дека проблемот (3.15) претставува инстанца на семидефинитното програмирање.

Формулацијата на проблемот (3.15) дозволува истиот да биде решен на нумерички доверлив и ефикасен начин. Впрочем, семидефинитните релаксации на разни квадратни оптимизациски проблеми можат да се совладаат лесно и ефикасно со користење на мноштво достапни софтверски пакети, како оптимизацискиот пакет CVX во MATLAB. Во позадина повеќето софтверски пакети за конвексна оптимизација ги напаѓаат семидефинитните програми со алгоритмите со внатрешна точка. Проблемот (3.15) може да биде решен со комплексност, во најлош случај, од

$$O(\max\{m,n\}^4\sqrt{n}\log(\frac{1}{\varepsilon})) \tag{3.16}$$

каде $\varepsilon > 0$. е точноста на решението. Ваквата комплексност (3.16) не ги зема во предвид специјалните структури, како на пример реткоста на матриците $\boldsymbol{C}, \boldsymbol{A}_1, ..., \boldsymbol{A}_m$. Постојат алгоритми кои ги искористуваат овие специјални структури за да го забрзаат процесот на решавање на проблемите. Пример за таков алгоритам е алгоритамот SeDuMi кој е еден од главните алгоритми за решавање во оптимизацискиот пакет CVX. За некои видови на специјални семидефинитни релаксации се развиваат специјално определени алгоритми кои имаат помала комплексност, [1].

Главен проблем кој настанува кога се користи методот на семидефинитна релаксација е претворањето на глобалното оптимално решение $\boldsymbol{X}^*$ на проблемот (3.15) во погодно решение $\tilde{\boldsymbol{x}}$ на проблемот (3.2). Во случај кога рангот на матрицата $\boldsymbol{X}^*$ е еднаков на 1, $\boldsymbol{X}^*$ може да се запише како $\boldsymbol{x}^*\boldsymbol{x}^{*T}$, при што $\boldsymbol{x}^*$ ќе биде погодно и оптимално решение на проблемот (3.2). Кога рангот на матрицата $\boldsymbol{X}^*$ е различен од 1, треба на ефикасен начин да се добие вектор $\tilde{\boldsymbol{x}}$ кој ќе биде погодно решение на проблемот (3.2). Постојат многу начини да се направи ова, меѓу кои се и методот на апроксимација со сопствени вектори и методот на рандомизација. Важно е да се напомене дека решението $\tilde{\boldsymbol{x}}$ добиено преку некоја од овие методи често не е оптималното решение.

3.1 Метод на апроксимација со сопствени вектори (Ранк-1 апроксимација)

Пред да го изложиме овој метод, ќе дадеме краток осврт на потребните поими од линеарна алгебра. Скаларот $\lambda \in \mathbb{R}$ е *сопствена вредност* на квадратната матрица $\boldsymbol{A}$ ако постои вектор $\boldsymbol{q} \neq \boldsymbol{0}$ за кој ќе биде исполнето

$$\boldsymbol{Aq} = \lambda\boldsymbol{q}. \tag{3.1.1}$$

Векторот $\boldsymbol{q}$ се нарекува *сопствен вектор* на матрицата $\boldsymbol{A}$. Нека матрицата $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ е позитивно семидефинитна матрица. Сите сопствени вектори на матрицата $\boldsymbol{P}$ соодветно можат да се запишат во колоните на квадратна матрица $\boldsymbol{Q}$. Слично, сопствените вредности на $\boldsymbol{P}$ соодветно можат да се запишат во главната дијагонала во дијагонална матрица $\boldsymbol{\Lambda}$. Во тој случај може да се запише

$$\boldsymbol{PQ} = \boldsymbol{Q\Lambda}. \tag{3.1.2}$$

Според тоа, матрицата $\boldsymbol{P}$ може да се изрази како

$$\boldsymbol{P} = \boldsymbol{Q\Lambda Q}^{-1}. \tag{3.1.3}$$

Сопствените вектори на реална симетрична матрица кои одговараат на различни сопствени вредности се меѓусебно ортогонални, [6], па за матрицата $\boldsymbol{Q}$ ќе биде исполнето

$$\boldsymbol{QQ}^{T} = \boldsymbol{I}. \tag{3.1.4}$$

Односно, важи дека матрицата $\boldsymbol{Q}$ е ортогонална матрица и важи:

$$\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{T}. \tag{3.1.5}$$

Релацијата (3.1.3) ја добива следната еквивалентна форма:

$$\boldsymbol{P} = \boldsymbol{Q\Lambda Q}^{T}. \tag{3.1.6}$$

Релацијата (3.1.6) претставува *декомпозиција со сопствени вредности и вектори* (анг. eigen-decomposition) на позитивно семидефинитната матрица $\boldsymbol{P}$, [7].

Овој вид декомпозиција може да се примени на матрицата $\boldsymbol{X}^{*}$, која е позитивно семидефинитна матрица со произволен ранк $r = rank(\boldsymbol{X}^{*})$:

$$\boldsymbol{X}^{*} = \sum_{i=1}^{r} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i^{T}, \tag{3.1.7}$$

каде што $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_r > 0$ се сопствените вредности на $\boldsymbol{X}^{*}$, а $\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_r \in \mathbb{R}^n$ се соодветните сопствени вектори. Целта на методот на апроксимација со сопствени вредности е матрицата $\boldsymbol{X}^{*}$ да се апроксимира со матрица $\boldsymbol{X}_1^{*}$ чиј ранг ќе биде еднаков на 1. Најдобрата таква апроксимација е дадена со

$$\boldsymbol{X}_1^{*} = \lambda_1 \boldsymbol{q}_1 \boldsymbol{q}_1^{T}. \tag{3.1.8}$$

Во овој случај можеме да го дефинираме

$$\tilde{\boldsymbol{x}} = \sqrt{\lambda_1}\,\boldsymbol{q}_1 \tag{3.1.9}$$

како можно решение на проблемот (3.2). Во случај кога решението (3.1.9) не е погодно решение на проблемот (3.2), решението (3.1.9) се мапира во погодно решение $\hat{\boldsymbol{x}}$ што се наоѓа во близина на $\tilde{\boldsymbol{x}}$, [1].

3.2 Метод на рандомизација

Нека $\boldsymbol{X} \in S^n$ е произволна симетрична, позитивно семидефинитна матрица. Се избира случаен вектор $\boldsymbol{\xi} \in \mathbb{R}^n$ со Гаусова распределба, средна вредност еднаква на 0 и коваријансна матрица $\boldsymbol{X}$, што пократко може да се запише како $\boldsymbol{\xi} \sim N(0, \boldsymbol{X})$. Идејата кај методот на рандомизација лежи во следната стохастичка квадратна програма со квадратни ограничувања:

$$\begin{aligned} &\min_{\boldsymbol{X}\in\mathbb{S}^n} \; E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}^T\boldsymbol{C}\boldsymbol{\xi}\right] \\ &s.t. \quad E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}^T\boldsymbol{A}_i\boldsymbol{\xi}\right]\trianglerighteq_i b_i, \quad i=1,...,m \\ &\qquad \boldsymbol{X}\succeq 0. \end{aligned} \tag{3.2.1}$$

Во проблемот (3.2.1) се манипулира со коваријансата матрица $\boldsymbol{X}$ на $\boldsymbol{\xi}$ со цел минимизирање на целната квадратна функција и задоволување на соодветните ограничувања. Коваријансната матрица $\boldsymbol{X}$ на случајниот вектор $\boldsymbol{\xi}$ се дефинира на следниот начин:

$$\boldsymbol{X} = E_{\xi\sim N(0,\boldsymbol{X})}\left[(\boldsymbol{\xi} - E_{\xi\sim N(0,\boldsymbol{X})}[\boldsymbol{\xi}])(\boldsymbol{\xi} - E_{\xi\sim N(0,\boldsymbol{X})}[\boldsymbol{\xi}])^T\right]. \tag{3.2.2}$$

Со оглед на тоа дека средната вредност на векторот $\boldsymbol{\xi}$ е еднаква на 0, релацијата (3.2.2) ја добива следната форма:

$$\boldsymbol{X} = E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}\boldsymbol{\xi}^T\right]. \tag{3.2.3}$$

Преку воочување на равенствата:

$$\begin{aligned} &E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}^T\boldsymbol{C}\boldsymbol{\xi}\right] = E_{\xi\sim N(0,\boldsymbol{X})}\left[Tr(\boldsymbol{\xi}^T\boldsymbol{C}\boldsymbol{\xi})\right] = E_{\xi\sim N(0,\boldsymbol{X})}\left[Tr(\boldsymbol{C}\boldsymbol{\xi}\boldsymbol{\xi}^T)\right] \\ &= Tr(E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{C}\boldsymbol{\xi}\boldsymbol{\xi}^T\right]) = Tr(\boldsymbol{C}E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}\boldsymbol{\xi}^T\right]) = Tr(\boldsymbol{C}\boldsymbol{X}), \\ &E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}^T\boldsymbol{A}_i\boldsymbol{\xi}\right] = E_{\xi\sim N(0,\boldsymbol{X})}\left[Tr(\boldsymbol{\xi}^T\boldsymbol{A}_i\boldsymbol{\xi})\right] = E_{\xi\sim N(0,\boldsymbol{X})}\left[Tr(\boldsymbol{A}_i\boldsymbol{\xi}\boldsymbol{\xi}^T)\right] \\ &= Tr(E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{A}_i\boldsymbol{\xi}\boldsymbol{\xi}^T\right]) = Tr(\boldsymbol{A}_i E_{\xi\sim N(0,\boldsymbol{X})}\left[\boldsymbol{\xi}\boldsymbol{\xi}^T\right]) = Tr(\boldsymbol{A}_i\boldsymbol{X}), \end{aligned} \tag{3.2.4}$$

[8], добиваме дека стохастичкиот квадратен проблем со квадратни ограничувања (3.2.1) е еквивалентен на проблемот (3.15).

Стохастичкиот квадратен проблем со квадратни ограничувања (3.2.1) претставува интерпретација на семидефинитната релаксација која овозможува алтернативен начин на генерирање на приближни решенија на проблемот (3.2). По добивањето на оптимално решение $\boldsymbol{X}^*$ на проблемот (3.15), се генерираат случајни вектори $\boldsymbol{\xi} \sim N(0, \boldsymbol{X}^*)$ кои се користат за конструирање на приближно решение на проблемот (3.2).

3.3 Лагранжов бидуален проблем

Проблем од класата на квадратни програми со квадратни ограничувања може да се формулира и како

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f_0(x) := \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} \\ &s.t. \quad f_i(x) := \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x} + b_i = 0, \quad i=1,...,m, \end{aligned} \tag{3.3.1}$$

каде матриците $\boldsymbol{C}, \boldsymbol{A}_1, ..., \boldsymbol{A}_m \in \mathbb{S}^n$, а скаларите $b_1, ..., b_m \in \mathbb{R}$. Иако досега во дипломската работа ја дефиниравме и опишавме само интерпретацијата на семидефинитната релаксација со релаксација на ограничувањето со рангот, проблемот (3.3.1) има пар семидефинитни релаксации кои се меѓусебно дуални. *Примарната семидефинитна релаксација* на проблемот (3.3.1) е

$$\begin{aligned} &\min_{\boldsymbol{X}\in\mathbb{S}^n} \; Tr(\boldsymbol{CX}) \\ &s.t. \;\; Tr(\boldsymbol{A}_i\boldsymbol{X})+b_i=0, \quad i=1,...,m \\ &\qquad \boldsymbol{X}\succeq 0. \end{aligned} \tag{3.3.2}$$

Веќе се уверивме дека постои бијекција помеѓу решенијата на (3.3.1) и решенијата со ранг 1 на (3.3.2), која е дадена преку:

$$\boldsymbol{x}\mapsto \boldsymbol{x}\boldsymbol{x}^T. \tag{3.3.3}$$

Со други зборови, утврдивме дека доколку оптималното решение на (3.3.2) има ранг кој е еднаков на 1, тогаш од тоа решение може да се изведе и оптималното решение на (3.3.1). Примарната семидефинитна релаксација на проблемот (3.3.1) е позната и под името *релаксација на Шор* (анг. Shor).

Дополнително, може да се изведе уште една релаксација на (3.3.1):

$$\begin{aligned} &\max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \; \sum_{i=1}^{m}\lambda_i b_i \\ &s.t. \;\; H(\boldsymbol{\lambda})\succeq 0, \end{aligned} \tag{3.3.4}$$

каде што $\boldsymbol{\lambda}\in\mathbb{R}^m$ се Лагранжовите множители на $\boldsymbol{x}$, а $H(\boldsymbol{\lambda})\in\mathbb{S}^n$ е половина од Хесеовата матрица на Лангранжовата функција $L(\boldsymbol{x},\boldsymbol{\lambda})$:

$$L(\boldsymbol{x},\boldsymbol{\lambda}) := f_0(\boldsymbol{x})+\sum_{i=1}^{m}\lambda_i f_i(\boldsymbol{x}), \tag{3.3.5}$$

$$H(\boldsymbol{\lambda}) := \frac{1}{2}\nabla^2_{\boldsymbol{xx}}L(\boldsymbol{x},\boldsymbol{\lambda}) = \boldsymbol{C}+\sum_{i=1}^{m}\lambda_i\boldsymbol{A}_i. \tag{3.3.6}$$

Проблемот (3.3.4) е *дуален семидефинитен проблем* на проблемот (3.3.2) и *Лагранжов дуален проблем* на проблемот (3.3.1). Неговата целна функција традиционално може да се запише, [9], како

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^m}\min_{\boldsymbol{x}\in\mathbb{R}^n} L(\boldsymbol{x},\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} g(\boldsymbol{\lambda}). \tag{3.3.7}$$

4 Проширување на генерални случаи

Методот на семидефинитна релаксација наоѓа примена и на проблеми надвор од множеството на реални хомогени квадратни програми со квадратни ограничувања. Техниката може да се искористи и во решавањето на нехомогени квадратни програми, комплексни проблеми и раздвоиви квадратни програми со квадратни ограничувања, кои ќе бидат илустрирани во продолжение.

4.1 Нехомогени проблеми

За еден квадратен оптимизациски проблем со квадратни ограничувања се вели дека е *хомоген* доколку нема линеарни членови. Досега во дипломската работа сите главни резултати беа формулирани за хомогени квадратни оптимизациски програми со квадратни ограничувања. Меѓутоа, проблемите кои се појавуваат во пракса се често

нехомогени, односно имаат линеарни членови во целната и ограничувачките функции. Во оваа класа на проблеми припаѓаат проблемите од видот

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}+2\boldsymbol{c}^T\boldsymbol{x}+d_0 \\ &s.t. \quad \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x}+2\boldsymbol{a}_i^T\boldsymbol{x}+d_i \trianglerighteq_i b_i, \quad i=1,...,m. \end{aligned} \tag{4.1.1}$$

каде што $\boldsymbol{C},\boldsymbol{A}_1,...,\boldsymbol{A}_i\in\mathbb{R}^{n\times n}$, $\boldsymbol{c},\boldsymbol{a}_1,...,\boldsymbol{a}_m\in\mathbb{R}^n$ и $d_0,d_1,...,d_m,b_1,...,b_m\in\mathbb{R}$. За ослободување од линеарните членови се воведува *хомогенизирачка променлива* $t\in\mathbb{R}$, таква што $t^2=1$. Нехомогената квадратна целна фукнција $f_0(x)=\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}+2\boldsymbol{c}^T\boldsymbol{x}+d_0$ може да се хомогенизира на следниот начин:

$$\begin{aligned} &\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}+2t\boldsymbol{c}^T\boldsymbol{x}+t^2d_0=\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x}+t\boldsymbol{x}^T\boldsymbol{c}+t\boldsymbol{c}^T\boldsymbol{x}+t^2d_0= \\ &=\begin{bmatrix}\boldsymbol{x}^T & t\end{bmatrix}\begin{bmatrix}\boldsymbol{C} & \boldsymbol{c}\\ \boldsymbol{c}^T & d_0\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix}=\boldsymbol{y}^T\boldsymbol{G}\boldsymbol{y}\ , \end{aligned} \tag{4.1.2}$$

каде што $\boldsymbol{y}:=\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix}\in\mathbb{R}^{n+1}$, а матрицата $\boldsymbol{G}:=\begin{bmatrix}\boldsymbol{C} & \boldsymbol{c}\\ \boldsymbol{c}^T & d_0\end{bmatrix}\in\mathbb{S}^{n+1}$. Слично се постапува и со ограничувачките функции:

$$\begin{aligned} &\boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x}+2t\boldsymbol{a}_i^T\boldsymbol{x}+t^2d_i=\boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{x}+t\boldsymbol{x}^T\boldsymbol{a}_i+t\boldsymbol{a}_i^T\boldsymbol{x}+t^2d_i= \\ &=\begin{bmatrix}\boldsymbol{x}^T & t\end{bmatrix}\begin{bmatrix}\boldsymbol{A}_i & \boldsymbol{a}_i\\ \boldsymbol{a}_i^T & d_i\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix}=\boldsymbol{y}^T\boldsymbol{H}_i\boldsymbol{y}, \end{aligned} \tag{4.1.3}$$

каде што матриците $\boldsymbol{H}_i:=\begin{bmatrix}\boldsymbol{A}_i & \boldsymbol{a}_i\\ \boldsymbol{a}_i^T & d_i\end{bmatrix}\in\mathbb{S}^{n+1}$ за $i=1,...,m$. Според ова проблемот (4.1.1) се трансформира во:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n,t\in\mathbb{R}} \begin{bmatrix}\boldsymbol{x}^T & t\end{bmatrix}\begin{bmatrix}\boldsymbol{C} & \boldsymbol{c}\\ \boldsymbol{c}^T & d_0\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix} \\ &s.t. \quad t^2=1, \\ &\qquad \begin{bmatrix}\boldsymbol{x}^T & t\end{bmatrix}\begin{bmatrix}\boldsymbol{A}_i & \boldsymbol{a}_i\\ \boldsymbol{a}_i^T & d_i\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix}\trianglerighteq_i b_i, \quad i=1,...,m\ , \end{aligned} \tag{4.1.4}$$

односно:

$$\begin{aligned} &\min_{\boldsymbol{y}\in\mathbb{R}^{n+1}} \ \boldsymbol{y}^T\boldsymbol{G}\boldsymbol{y} \\ &s.t. \quad y_{n+1}^2=1, \\ &\qquad \boldsymbol{y}^T\boldsymbol{H}_i\boldsymbol{y}\trianglerighteq_i b_i, \quad i=1,...,m. \end{aligned} \tag{4.1.5}$$

Проблемот (4.1.5) е реална хомогена квадратна програма со квадратни ограничувања, чиишто големина и број на ограничувања се за 1 поголеми од оние на проблемот (4.1.1). Според тоа, методот на семидефинитна релаксација може да се употреби и во решавањето на нехомогени проблеми преку нивна претходна хомогенизација. $\boldsymbol{x}^*$ е оптимално решение на нехомогената квадратна програма со квадратни ограничувања

(4.1.1) ако и само ако $\begin{bmatrix} \boldsymbol{x}^* \\ 1 \end{bmatrix}$ или $\begin{bmatrix} -\boldsymbol{x}^* \\ -1 \end{bmatrix}$ се решенија на неговата хомогенизирана форма (4.1.5), [9].

Нехомоген квадратен проблем кој често се јавува во практичните проблеми е проблемот на најмали квадрати со ограничувања. Пример за таков проблем е

$$\min_{\boldsymbol{x}\in\{\pm 1\}^n} \left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 , \tag{4.1.6}$$

каде $\boldsymbol{A}\in\mathbb{R}^{k\times n}$ $(k\geq n)$, $\boldsymbol{x}\in\{\pm 1\}^n$ и $\boldsymbol{b}\in\mathbb{R}^k$. Целната функција може да се запише како

$$\begin{aligned} \left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 &= (a_{11}x_1+\ldots+a_{1n}x_n-b_1)^2+\ldots+(a_{k1}x_1+\ldots+a_{kn}x_n-b_k)^2 = \\ &= \sum_{i=1}^{k}(\boldsymbol{a}_i^T\boldsymbol{x}-b_i)^2 , \end{aligned} \tag{4.1.7}$$

каде $\boldsymbol{a}_i^T$ ја претставува i-тата редица на матрицата $\boldsymbol{A}$. Поради линеарните членови што ќе се појават во (4.1.7), јасно е дека целната функција ќе биде нехомогена функција. Со воведување на хомогенизирачка променлива $t\in\mathbb{R}$, таква што $t^2=1$, целната функција можеме да ја хомогенизираме на следниот начин:

$$\begin{aligned} \left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 = \left\|\boldsymbol{A}\boldsymbol{x}-t\boldsymbol{b}\right\|_2^2 = \sum_{i=1}^{k}(\boldsymbol{a}_i^T\boldsymbol{x}-tb_i)^2 = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-t\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x}-t\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b}+t^2\left\|\boldsymbol{b}\right\|_2^2 = \\ = \begin{bmatrix} \boldsymbol{x}^T & t \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^T\boldsymbol{A} & -\boldsymbol{A}^T\boldsymbol{b} \\ -\boldsymbol{b}^T\boldsymbol{A} & \left\|\boldsymbol{b}\right\|_2^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix}. \end{aligned} \tag{4.1.8}$$

Според тоа, проблемот (4.1.6) може да се запише како

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n,\, t\in\mathbb{R}} \ & \begin{bmatrix} \boldsymbol{x}^T & t \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^T\boldsymbol{A} & -\boldsymbol{A}^T\boldsymbol{b} \\ -\boldsymbol{b}^T\boldsymbol{A} & \left\|\boldsymbol{b}\right\|_2^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \\ s.t. \quad & t^2=1, \\ & x_i^2=1, \quad i=1,\ldots,n. \end{aligned} \tag{4.1.9}$$

Проблемот (4.1.9) е хомоген реален квадратен оптимизациски проблем со квадратни ограничувања и на него може да се примени методот на семидефинитна релаксација.

4.2 Комплексни проблеми

Во комплексни хомогени квадратни програми со квадратни ограничувања припаѓаат проблемите од видот:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{C}^n} \ & \boldsymbol{x}^H\boldsymbol{C}\boldsymbol{x} \\ s.t. \quad & \boldsymbol{x}^H\boldsymbol{A}_i\boldsymbol{x} \trianglerighteq_i b_i, \quad i=1,\ldots,m , \end{aligned} \tag{4.2.1}$$

каде што матриците $\boldsymbol{C}, \boldsymbol{A}_1, ..., \boldsymbol{A}_m \in H^n$ (H^n го претставува множеството на сите комплексни $n \times n$ ермитски[4] матрици). Со примена на истата идеја како во проблемите со реални вредности на променливите, се добива следната семидефинитна релаксација на проблемот (4.2.1):

$$\begin{aligned} &\min_{\boldsymbol{X} \in H^n} \; Tr(\boldsymbol{C}\boldsymbol{X}) \\ &s.t. \quad Tr(\boldsymbol{A}_i \boldsymbol{X}) \trianglerighteq_i b_i, \quad i = 1, ..., m \\ &\qquad \boldsymbol{X} \succeq 0. \end{aligned} \tag{4.2.2}$$

Пример за комплексен квадратен проблем кој се среќава во пракса е дискретниот квадратен оптимизациски проблем во комплексна ермитска форма

$$\begin{aligned} &\max_{\boldsymbol{x} \in \mathbb{C}^n} \; \boldsymbol{x}^H \boldsymbol{Q} \boldsymbol{x} \\ &s.t. \quad x_i \in \{1, e^{\frac{j2\pi}{M}}, ..., e^{\frac{j2\pi(M-1)}{M}}\}, \quad i = 1, ..., n, \end{aligned} \tag{4.2.3}$$

каде што $\boldsymbol{Q} \in \mathbb{C}^{n \times n}$ е ермитска матрица, односно $\boldsymbol{Q} \in H^n$. За да се опише постапката на изведување на семидефинитната релаксација на проблемот (4.2.3) се дефинира вектор

$$\boldsymbol{v}_i = \overline{x}_i \boldsymbol{u}, \quad i = 1, ..., n, \tag{4.2.4}$$

за некој произволен $\boldsymbol{u} \in \mathbb{C}^n$, за кој важи $\|\boldsymbol{u}\|_2^2 = 1$. Проблемот (4.2.3) може да се запише во следната форма:

$$\begin{aligned} &\max_{\boldsymbol{v}_i \in \mathbb{C}^n} \; \sum_{l=1}^{n} \sum_{k=1}^{n} \boldsymbol{v}_k^H \boldsymbol{v}_l Q_{lk} \\ &s.t. \quad \boldsymbol{v}_i = \overline{x}_i \boldsymbol{u}, \quad i = 1, ..., n, \\ &\qquad \|\boldsymbol{u}\|_2^2 = 1 \\ &\qquad x_i \in \{1, e^{\frac{j2\pi}{M}}, ..., e^{\frac{j2\pi(M-1)}{M}}\}, \quad i = 1, ..., n. \end{aligned} \tag{4.2.5}$$

Бидејќи $\|\boldsymbol{u}\|_2^2 = 1$ и $\|x_i\|_2^2 = 1$, следува дека решенијата $\boldsymbol{v}_i$ на проблемот (4.2.5) исто така имаат единична Евклидска норма, односно $\|\boldsymbol{v}_i\|_2^2 = 1$ за $i = 1, ..., n$. Според ова, ограничувањата на проблемот (4.2.5) можат да се релаксираат и истиот ја добива следната форма:

$$\begin{aligned} &\max_{\boldsymbol{v}_i \in \mathbb{C}^n} \; \sum_{l=1}^{n} \sum_{k=1}^{n} \boldsymbol{v}_k^H \boldsymbol{v}_l Q_{lk} \\ &s.t. \quad \|\boldsymbol{v}_i\|_2^2 = 1, \quad i = 1, ..., n. \end{aligned} \tag{4.2.6}$$

---

[4] Ермитската матрица $\boldsymbol{A}$ е комплексна квадратна матрица која е еднаква на својата конјугирано транспонирана матрица, односно

$$\boldsymbol{A} = \overline{\boldsymbol{A}^T} = \boldsymbol{A}^H$$

Дефинираме матрица $\boldsymbol{V} = [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \ldots \ \boldsymbol{v}_n]$ и матрица $\boldsymbol{X} = \boldsymbol{V}^H \boldsymbol{V}$. Матрицата $\boldsymbol{X}$ е ермитска позитивно семидефинитна матрица. Дополнително, бидејќи $\boldsymbol{v}_k^H \boldsymbol{v}_l = X_{kl}$, целната функција во (4.2.6) може да се изрази како $Tr(\boldsymbol{XQ})$. Конечно, се добива

$$\begin{aligned} &\max_{X \in H^n} \ Tr(\boldsymbol{XQ}) \\ &s.t. \quad X_{ii} = 1, \quad i = 1, \ldots, n, \\ &\qquad \boldsymbol{X} \succeq 0 . \end{aligned} \tag{4.2.7}$$

Проблемот (4.2.3) наоѓа примена во детекцијата на M-арни PSK повеќекорисnички сигнали, [10].

4.3 Раздвоиви квадратни програми со квадратни ограничувања

Раздвоиви квадратни програми со квадратни ограничувања се проблемите кои ја имаат следната форма:

$$\begin{aligned} &\min_{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{C}^n} \ \sum_{i=1}^{k} \boldsymbol{x}_i^H \boldsymbol{C}_i \boldsymbol{x}_i \\ &s.t. \qquad \sum_{l=1}^{k} \boldsymbol{x}_l^H \boldsymbol{A}_{i,l} \boldsymbol{x}_l \trianglerighteq_i b_i, \quad i = 1, \ldots, m. \end{aligned} \tag{4.3.1}$$

Нека $\boldsymbol{X}_i = \boldsymbol{x}_i \boldsymbol{x}_i^T$, за $i = 1, \ldots, k$. Со релаксирање на ограничувањето за рангот на секое $\boldsymbol{X}_i$ ја добиваме следната семидефинитна релаксација на проблемот (4.3.1):

$$\begin{aligned} &\min_{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k \in H^n} \ \sum_{i=1}^{k} Tr(\boldsymbol{C}_i \boldsymbol{X}_i) \\ &s.t. \qquad \sum_{l=1}^{k} Tr(\boldsymbol{A}_{i,l} \boldsymbol{X}_l) \trianglerighteq_i b_i, \quad i = 1, \ldots, m, \\ &\qquad \boldsymbol{X}_1 \succeq 0, \ldots, \boldsymbol{X}_k \succeq 0. \end{aligned} \tag{4.3.2}$$

5 MIMO детекција

Multiple-input multiple-output (MIMO) е безжична технологија која користи повеќе предавателни и приемни антени, со цел зголемување на податочните брзини преку мултиплексирање, како и подобрување на перформансите на системот преку диверзитет. MIMO комуникациските системи се појавуваат во многу модерни безжични телекомуникациски стандарди, почнувајќи од IEEE 802.11n (WiFi 4) и HSPA+ (3G). Во споредба со традиционалните single-input single-output телекомуникациски системи, користењето на повеќе предавателни и повеќе приемни антени носи значителни придобивки во функционирањето на безжичниот систем. За да се искористат овие придобивки, приемникот треба со максимална енергетска ефикасност и минимално зголемување на севкупната комплексност да ги детектира пренесените симболи.

Како што Клод Шенон појаснил во својата позната статија “Математичка теорија на телекомуникациите”, основниот проблем во телекомуникациите е точното или приближно репродуцирање во една точка на порака која била избрана во друга точка. Со други зборови, проблемот кој се среќава во дизајнирањето на приемници за дигиталните телекомуникациски системи е точното детектирање на испратените

симболи преку нивните зашумени приемни мерења. Во сите реални сценарија поради присуството на шумот, приемникот ќе прави повремени грешки во одлучувањето. Токму затоа била и сè уште е привлечна идејата да се направи приемник кој има својство да ја минимизира веројатноста за грешка.

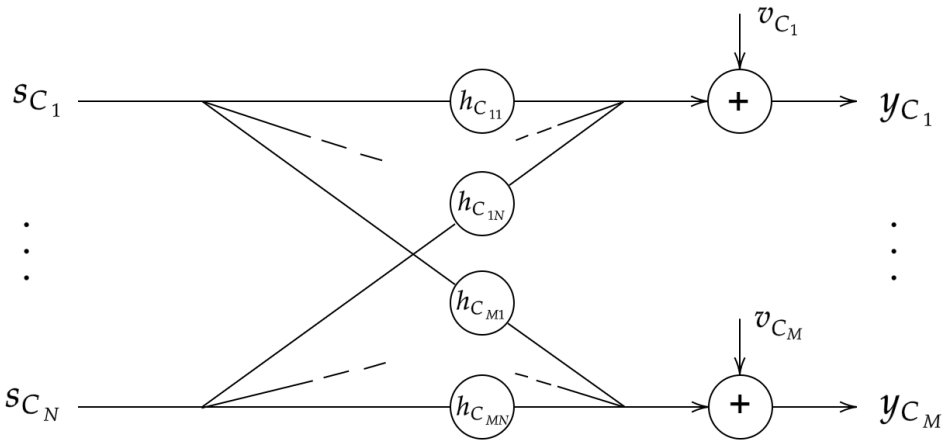
Кај MIMO системите истовремено се пренесуваат повеќе симболи, на кои попатно влијаат феномени како случаен шум и меѓуканална интерференција. Во приемникот овие симболи можат да бидат детектирани поединечно или заедно. Истовремената детекција на повеќе симболи во MIMO системите е од централна важност за целосно искористување на придобивките кои ги нудат MIMO техниките. Проблемот на истовремено оптимално детектирање на симболите, во смисла на минимизирање на вкупната веројатност за грешка се решава со помош на ML (Maximum Likelihood) детектор. За жал, со ваквата детекција проблемот е NP-тежок, што значи дека сите алгоритми создадени за негово оптимално решавање имаат комплексност која експоненцијално се зголемува со зголемувањето на бројот на симболи кои се детектираат заедно. Со други зборови, доколку не постои некаква специјална структура на проблемот, не постојат алгоритми за негово решавање кои имаат полиномна комплексност, [11].

Поради наведените причини, во MIMO системите се користатат голем број на суб-оптимални или квази-оптимални приемници, кои имаат помала пресметковна комплексност. Најголемата класа на суб-оптимални детектори е класата на линеарни детектори, кои вршат линеарна трансформација на примените симболи пред одлучувањето. Најпознати детектори од овој вид се ZF (zero-forcing) и MMSE (minimum mean square error) детекторите. Понапредна класа на детектори е класата на DF (decision feedback) детекторите, кои одлучуваат симбол по симбол и го одземаат влијанието на моменталниот симбол врз останатите симболи пред наредното одлучување. Нормално, постои компромис помеѓу пресметковната комплексност и точноста на погорните детектори. Линеарните детектори се најбрзи, но имаат полоши перформанси од DF детекторите, кои се пресметковно покомплексни, [12].

Една од подобрите квази-оптимални MIMO детекциски стратегии е токму семидефинитната релаксација, која обезбедува помала веројатност за битски грешки од линеарните и DF детекторите. Семидефинитната релаксација го апроксимира решавањето на ML проблемот со конвексна програма, која може ефикасно да се реши во полиномно време, [11]. Пристапот на семидефинитната релаксација е прво да го формулира ML проблемот во поголема димензија, а потоа да ги релаксира неконвексните ограничувања, што ќе го илустрираме во продолжение.

5.1 Формулација на проблемот

Линеарен MIMO канал е прикажан на сликата 5.1.1. Предавателот истовремено испраќа N симболи $s_{C_1}, s_{C_2}, ..., s_{C_N}$ кои припаѓаат на конечното множество или *констелацијата* $S_C \subset \mathbb{C}$. Во приемникот пристигнуваат M сигнали $y_{C_1}, y_{C_2}, ..., y_{C_M}$, кои претставуваат линеарна комбинација од N-те влезни симболи и адитивна компонента на шумот. Во општ случај, освен ако не е нагласено поинаку, се претпоставува дека бројот на приемни сигнали M е поголем од бројот на пренесени симболи N, односно $M \geq N$.



Слика 5.1.1 Линеарен MIMO канал со *N* влезови и *M* излези

Во случај на стандарден MIMO канал, приемниот сигнал може да се опише на следниот начин

$$\boldsymbol{y}_C = \boldsymbol{H}_C \boldsymbol{s}_C + \boldsymbol{v}_C , \tag{5.1.1}$$

каде матрицата $\boldsymbol{H}_C \in \mathbb{C}^{M\times N}$ е *каналната матрица*, а векторот $\boldsymbol{v}_C \in \mathbb{C}^M$ е *адитивниот шум*. Векторите $\boldsymbol{y}_C \in \mathbb{C}^M$ и $\boldsymbol{s}_C \in S^N$ соодветно ги означуваат приемните сигнали и пренесените симболи. Во примерот кој ќе го разгледуваме ќе претпоставиме дека пренесените симболи $\boldsymbol{s}_{C_i}$, за $i = 1,...,N$, следат кватернарна PSK (quaternary phase-shift-keying/QPSK) констелација, односно

$$S_C = \{1+j,\ \ -1+j,\ \ -1-j,\ \ 1-j\}. \tag{5.1.2}$$

ML детекторот го пребарува множеството на сите можни N-торки од симболи $\boldsymbol{s}_{C_i} \in S_C$, со цел да ги определи оние кои го минимизираат Евклидското растојание помеѓу приемниот сигнал $\boldsymbol{y}_C$ и реконструираниот сигнал $\boldsymbol{H}_C \boldsymbol{s}_C$. Со други зборови, ML проблемот е еквивалентен на дискретниот проблем на најмали квадрати со ограничувања:

$$\min_{\boldsymbol{s}_C \in \{\pm 1 \pm j\}^N} \left\| \boldsymbol{y}_C - \boldsymbol{H}_C \boldsymbol{s}_C \right\|_2^2 . \tag{5.1.3}$$

Моделот на системот е целосно дефиниран за комплексни вредности на променливите, но поради поголема едноставност во понатамошните пресметки се преминува во доменот на реални броеви. Се дефинираат матриците:

$$\boldsymbol{y} = \begin{bmatrix} \mathrm{Re}\{\boldsymbol{y}_C\} \\ \mathrm{Im}\{\boldsymbol{y}_C\} \end{bmatrix} \tag{5.1.4}$$

$$\boldsymbol{s} = \begin{bmatrix} \mathrm{Re}\{\boldsymbol{s}_C\} \\ \mathrm{Im}\{\boldsymbol{s}_C\} \end{bmatrix} \tag{5.1.5}$$

$$\boldsymbol{v} = \begin{bmatrix} \mathrm{Re}\{\boldsymbol{v}_C\} \\ \mathrm{Im}\{\boldsymbol{v}_C\} \end{bmatrix} \tag{5.1.6}$$

$$\boldsymbol{H} = \begin{bmatrix} \mathrm{Re}\{\boldsymbol{H}_C\} & -\mathrm{Im}\{\boldsymbol{H}_C\} \\ \mathrm{Im}\{\boldsymbol{H}_C\} & \mathrm{Re}\{\boldsymbol{H}_C\} \end{bmatrix} \tag{5.1.7}$$

Проблемот (5.1.3) ја добива следната форма:

$$\min_{\boldsymbol{s}\in\{\pm 1\}^{2N}} \ \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|_2^2 \ . \tag{5.1.8}$$

Проблемот (5.1.8) не претставува хомогена квадратна програма со квадратни ограничувања, но како што веќе илустриравме претходно во овој дипломски труд, тоа може да се среди со воведување на хомогенизирачка променлива $t \in \mathbb{R}$, т.ш. $t^2 = 1$:

$$\begin{aligned} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|_2^2 = \|t\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|_2^2 = \boldsymbol{s}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{s} - t\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{s} - t\boldsymbol{s}^T\boldsymbol{H}^T\boldsymbol{y} + t^2\|\boldsymbol{y}\|_2^2 = \\ = \begin{bmatrix} \boldsymbol{s}^T & t \end{bmatrix} \begin{bmatrix} \boldsymbol{H}^T\boldsymbol{H} & -\boldsymbol{H}^T\boldsymbol{y} \\ -\boldsymbol{y}^T\boldsymbol{H} & \|\boldsymbol{y}\|_2^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{s} \\ t \end{bmatrix} \end{aligned} \tag{5.1.9}$$

Сега, проблемот (5.1.8) може да се формулира на следниов начин:

$$\begin{aligned} &\min_{\boldsymbol{s}\in\mathbb{R}^{2N}, t\in\mathbb{R}} \ \|t\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|_2^2 \\ &s.t. \qquad s_i^2 = 1, \quad i = 1,\ldots,2N, \\ &\qquad\quad t^2 = 1, \end{aligned} \tag{5.1.10}$$

односно,

$$\begin{aligned} &\min_{\boldsymbol{s}\in\mathbb{R}^{2N}, t\in\mathbb{R}} \ \begin{bmatrix} \boldsymbol{s}^T & t \end{bmatrix} \begin{bmatrix} \boldsymbol{H}^T\boldsymbol{H} & -\boldsymbol{H}^T\boldsymbol{y} \\ -\boldsymbol{y}^T\boldsymbol{H} & \|\boldsymbol{y}\|_2^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{s} \\ t \end{bmatrix} \\ &s.t. \qquad s_i^2 = 1, \quad i = 1,\ldots,2N, \\ &\qquad\quad t^2 = 1. \end{aligned} \tag{5.1.11}$$

За поголема едноставност во понатамошните изведувања, нека

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{s} \\ t \end{bmatrix} \in \mathbb{R}^{2N+1}, \tag{5.1.12}$$

$$\boldsymbol{C} := \begin{bmatrix} \boldsymbol{H}^T\boldsymbol{H} & -\boldsymbol{H}^T\boldsymbol{y} \\ -\boldsymbol{y}^T\boldsymbol{H} & \|\boldsymbol{y}\|_2^2 \end{bmatrix} \in \mathbb{S}^{2N+1}. \tag{5.1.13}$$

Конечно, формулацијата на MIMO ML проблемот како реална хомогена квадратна програма со квадратни ограничувања е следна:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^{2N+1}} \ \boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} \\ &s.t. \qquad x_i^2 = 1, \quad i = 1,\ldots,2N+1. \end{aligned} \tag{5.1.14}$$

Формулацијата (5.1.14) претставува бинарна квадратна програма. Поврзаноста на оптималните решенија на хомогениот проблем (5.1.14) и на оригиналниот проблем

(5.1.8) е следна: Ако $\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{s}^* \\ t^* \end{bmatrix}$ е оптималното решение на (5.1.14), тогаш $\boldsymbol{s}^*$ е оптимално решение на (5.1.8) во случај кога $t^* = 1$, додека $-\boldsymbol{s}^*$ е оптимално решение кога $t^* = -1$.

Семидефинитната релаксација на проблемот (5.1.14) се изведува на начинот претходно опишан во овој дипломски труд, со воведување на нова променлива $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$:

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{S}^{2N+1}} \quad & Tr(\boldsymbol{C}\boldsymbol{X}) \\ s.t. \quad & X_{ii} = 1, \quad i = 1, ..., 2N \\ & \boldsymbol{X} \succeq 0, \qquad rank(\boldsymbol{X}) = 1, \end{aligned} \tag{5.1.15}$$

и отфрлање на ограничувањето $rank(\boldsymbol{X}) = 1$:

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{S}^{2N+1}} \quad & Tr(\boldsymbol{C}\boldsymbol{X}) \\ s.t. \quad & X_{ii} = 1, \quad i = 1, ..., 2N \\ & \boldsymbol{X} \succeq 0. \end{aligned} \tag{5.1.16}$$

Како што веќе претходно истакнавме, основен проблем кој се јавува е добивање на погодно решение $\tilde{\boldsymbol{x}}$ на проблемот (5.1.14) од глобалното оптимално решение $\boldsymbol{X}^*$ на конвексниот проблем (5.1.16). Во продолжение ќе преминеме на практично формулирање и решавање на ML проблемот со помош на софтверскиот пакет CVX во MATLAB.

5.2 Формулирање на MIMO ML проблемот во MATLAB и добивање на глобално оптимално решение $\boldsymbol{X}^*$

За почеток ги дефинираме параметрите на проблемот, како што е прикажано на сликата 5.2.1. Големината на проблемот е определена преку бројот на пренесени симболи N и бројот на приемни сигнали M . Односот сигнал-шум (анг. signal-to-noise ratio/SNR) го поставуваме на произволна вредност и од него ја пресметуваме варијансата на шумот σ_{snr} . Следен чекор е генерирање на случајни бити со помош на функцијата randi() и нивно претворање во QPSK симболи $\boldsymbol{s}_c$. Матрицата $\boldsymbol{H}_c$ се генерира како случајна матрица со големина $M \times N$. Векторот на приемните сигнали, приемниот вектор $\boldsymbol{y}_c$, се добива преку множење на каналната матрица $\boldsymbol{H}_c$ со $\boldsymbol{s}_c$ и додавање на адитивниот шум. Следно, се преминува во доменот на реалните броеви и се дефинираат векторот $\boldsymbol{y}$ и матрицата $\boldsymbol{H}$. Последно, се дефинира матрицата $\boldsymbol{C}$, добиена со погоре опишаната хомогенизација на проблемот.

```
% Големина на проблемот
M = 4;
N = 4;

% Однос сигнал-шум
SNR = 12;
sigma_snr = sqrt( N*10 ^ ( - (SNR) / 10 ) );

% Генерирање на битите и нивно претворање во QPSK симболи
s = randi(2,2*N,1)-1;
s = 2*s - 1;
s_c = (s(1:length(s)/2) + 1i*s(length(s)/2 + 1:end))/sqrt(2);

% Генерирање на каналната матрица H_c
H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));

% Генерирање на приемниот вектор y_c
y_c = H_c*s_c + (sigma_snr/sqrt(2))*(randn(M,1)+1i*randn(M,1));

% Премин од множеството на комплексни броеви во множеството на
% реални броеви
y = sqrt(2)*[real(y_c); imag(y_c)];
H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];

% Матрица C добиена со хомогенизација
C = [H'*H, -H'*y; -y'*H, norm(y)^2];
```

Слика 5.2.1 Дефинирање на параметрите на MIMO ML проблемот

Решавањето на оптимизацискиот проблем во MATLAB преку софтверскиот пакет CVX е прикажано на сликата 5.2.2. CVX претставува систем за конструирање и решавање на конвексни програми, кој подржува стандардни типови на проблеми како линеарни, квадратни и семидефинитни програми.

```
cvx_begin
variable X(2*N+1, 2*N+1) symmetric
minimize(trace(C*X));
subject to
for i=1:2*N+1
    % Ги поставуваме вредностите на главната дијагонала на матрицата X на
    % единици.
    X(i,i) == 1;
end;
X == semidefinite(2*N+1);
cvx_end
```

Слика 5.2.2 Решавање на MIMO ML проблемот преку софтверскиот пакет CVX во MATLAB

5.3 Решение со проверка на сите можности - стратегија „груба сила“ (анг. brute force/BF)

Пред да преминеме на методите преку кои ќе го екстрахираме погодното решение $\tilde{\boldsymbol{x}}$ на проблемот (5.1.14) од глобалното оптимално решение $\boldsymbol{X}^*$ на конвексниот проблем (5.1.16), ќе илустрираме едно решение со стратегијата „груба сила“ на оригиналниот проблем (5.1.8). Важно е да се напомене дека ова решение секогаш

точно го решава оптимизацискиот проблем, но е многу неефикасно, бидејќи ги проверува сите можности една по една.

Користејќи ги параметрите на проблемот дефинирани погоре, најпрвин се иницијализираат векторот $\boldsymbol{s}_{BF}$ и променливата MIN_BF на почетни вредности, како што е прикажано на сликата 5.3.1.

```
% Иницијализирање на векторот s_BF
s_BF = zeros(2*N,1) -1;

% Иницијализирање на променливата MIN_BF каде ќе се чува оптималната
% вредност на целната функција.
MIN_BF = norm(y-H*s_BF)^2;
```

Слика 5.3.1 Иницијализирање на s_BF и MIN_BF

Следен чекор е испитување на сите останати $2^{2N}-1$ можни вредности на векторот $\boldsymbol{s}_{BF}$ и пресметување на вредноста на целната функција $\left\|\boldsymbol{y}-\boldsymbol{H}\boldsymbol{s}_{BF}\right\|_2^2$ за секоја од нив. Доколку добиената вредност е помала од моменталната вредност зачувана во променливата MIN_BF, таа се поставува како нова вредност на MIN_BF. Целиот алгоритам е прикажан на сликата 5.3.2.

```
% Ги испитуваме сите можности.
% Итерираме од 1 до 2^(2N)-1
for i = 1:2^(2*N)-1
    % Вредноста на i ја претвораме во бинарен број и истиот го сместуваме
    % во матрицата binary (1 x 2N)
    binary = de2bi(i,2*N);
    % Вредноста на temp ја добиваме како вектор 2N x 1 наполнет со -1,
    % собран со вредноста на 2*binary'.
    temp = zeros(2*N,1) -1 + 2*binary';
    if norm(y-H*temp)^2 < MIN_BF
        MIN_BF = norm(y-H*temp)^2;
        s_BF = temp;
    end;
end;
```

Слика 5.3.2 Алгоритам за решение со стратегијата „груба сила“ на MIMO ML проблемот

5.4 Добивање на x^* преку методот на апроксимација со сопствени вектори (Ранк-1 апроксимација)

Откако сме го добиле глобалното оптимално решение $\boldsymbol{X}^*$ на конвексниот проблем (5.1.16) на начинот прикажан на сликата 5.2.2, следен чекор е од него да добиеме погодно решение $\tilde{\boldsymbol{x}}$ на проблемот (5.1.14). Претходно во оваа дипломска работа детално го објаснивме методот на апроксимација со сопствени вектори, каде се користи декомпозиција на матрицата $\boldsymbol{X}^*$ со сопствени вектори и вредности (анг. eigen-decompositon):

$$\boldsymbol{X}^* = \sum_{i=1}^{r} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i^T , \qquad (\mathbf{5.4.1})$$

каде што $r = rank(\boldsymbol{X}^*)$, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_r > 0$ се сопствените вредности на $\boldsymbol{X}^*$, а $\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_r \in \mathbb{R}^n$ се соодветните сопствени вектори. Можно решение $\tilde{\boldsymbol{x}}$ се добива како

$$\tilde{\boldsymbol{x}} = \sqrt{\lambda_1}\boldsymbol{q}_1, \qquad (\mathbf{5.4.2})$$

кое во случајов се мапира во погодно решение преку MATLAB функцијата sign(). На крај се извлекуваат вредностите за векторот $\tilde{\boldsymbol{s}}$ и според тие вредности се пресметува вредноста на целната функција на проблемот (5.1.8) која требаше да ја минимизираме. Имплементацијата на методот на апроксимација со сопствени вектори во MATLAB е прилично едноставна и е дадена на сликата 5.4.1.

```
% Разложување на сопствени вредности/вектори на X и определување на
% најголемата сопствена вредност Lambda_MAX и соодветниот сопствен вектор
% q.
[q, Lambda_MAX] = eigs(X,1);

% Можно решение
x_tilde_SDR_eigen = sqrt(Lambda_MAX)*q;

% Мапирање во погодно решение
x_tilde_SDR_eigen = sign(x_tilde_SDR_eigen);

% Извлекување на вредноста на хомогенизациската променлива t_tilde
t_tilde_SDR_eigen = x_tilde_SDR_eigen(2*N+1);

% Извлекување на вредноста на векторот s_tilde
if (t_tilde_SDR_eigen == -1)
    s_tilde_SDR_eigen = - x_tilde_SDR_eigen(1:2*N);
else
    s_tilde_SDR_eigen = x_tilde_SDR_eigen(1:2*N);
end

% Оптимална вредност
MIN_SDR_eigen = norm(y-H*s_tilde_SDR_eigen)^2;
```

Слика 5.4.1 Добивање на решение преку методот на апроксимација со сопствени вредности

5.5 Добивање на x^* преку методот на рандомизација

Постапката на Гаусова рандомизација за бинарна квадратна програма, каков што е MIMO ML проблемот, ќе биде изложена во продолжение. Тргнувајќи од оптималното глобално решение $\boldsymbol{X}^*$ на релаксираниот проблем, прв чекор е да одбереме вредност за параметарот L кој го означува бројот на рандомизации. Потоа се генерираат L случајни вектори $\boldsymbol{\xi}_l$ за $l = 1, ..., L$ со средна вредност 0 и коваријансна матрица $\boldsymbol{X}^*$. За секој од овие вектори се пресметува вредноста на целната функција во (5.1.14). Векторот $\boldsymbol{\xi}_l$ за кој ќе се добие најмала вредност на целната функција е апроксимираното решение $\tilde{\boldsymbol{x}}$ на (5.1.14). Често е потребно мапирање на избраниот вектор $\boldsymbol{\xi}_l$ во погодно решение $\tilde{\boldsymbol{x}}$ на (5.1.14), кое во случајов може да се направи со помош на функцијата sign во MATLAB. Ова мапирање може да се направи и на почеток при самото генерирање на $\boldsymbol{\xi}_l$, како што е направено во извадокот од кодот прикажан на сликата 5.5.1.

```
% Генерирање на L случајни вектори со средна вредност 0 и коваријансна
% матрица X
xi_l = sign(mvnrnd(zeros(2*N+1,1),X,L));

% Иницијализирање на векторот x_tilde
x_tilde_SDR_rand = xi_l(1,:)';

% Иницијализирање на променливата MIN_rand
MIN_rand = xi_l(1,:) * C * xi_l(1,:)';

for i = 2:L
    % Моменталната вредност на целната функција
    temp = xi_l(i,:) * C * xi_l(i,:)';
    if temp < MIN_rand
        x_tilde_SDR_rand = xi_l(i,:)';
        MIN_rand = temp;
    end
end
```

Слика 5.5.1 Добивање на решение преку методот на рандомизација

Слично како и при решавањето со методот на апроксимација со сопствени вектори и овде на крај ги извлекуваме вредностите на векторот $\tilde{s}$, кој ги содржи одлучените реални вредности на испратените симболи. На крај за овие вредности се пресметува целната функција на MIMO ML проблемот (5.1.8) која требаше да се минимизира.

5.6 Нумерички резултати

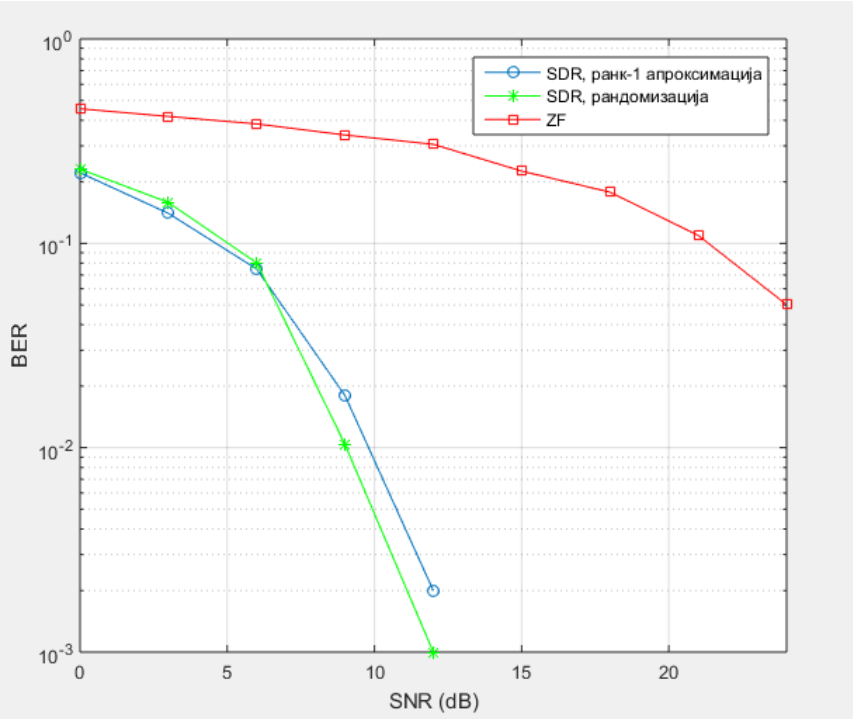
Во продолжение ќе ги изложиме резултатите од направените симулации кои ги потврдуваат добрите перформанси на MIMO детекторот со семидефинитна релаксација (SDR). Како што споменавме погоре, квази-оптималната MIMO детекција со семидефинитна релаксација обезбедува помала веројатност за битска грешка од линеарните и DF детектори задржувајќи полиномна комплексност.

5.6.1 BER перформанси

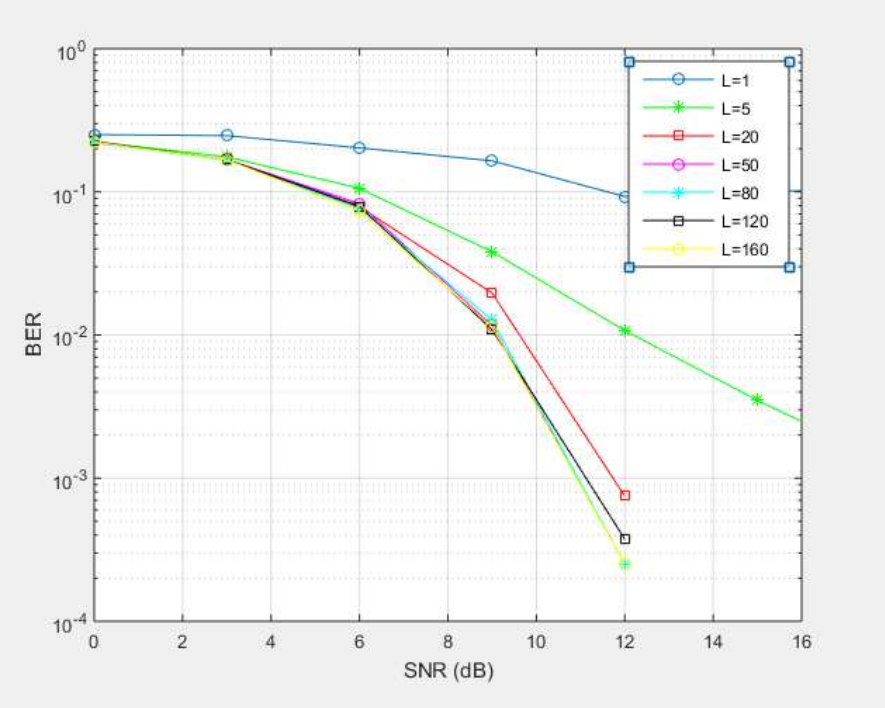
Во изведената симулација вршиме споредба на веројатностите за битска грешка (анг. Bit error/BER) на SDR детектор со методот на апроксимација со сопствени вектори (ранк-1 апроксимација), SDR детектор со методот на рандомизација (со $L = 50$ рандомизации) и ZF детектор (претставник од класата на линеарни детектори) за различни вредности на односот сигнал-шум. Модулациската шема која се користи е истата од погоре, кваратернарна PSK (QPSK), додека големината на проблемот, бројот на пренесени симболи и приемни сигнали е $(M, N) = (40, 40)$. Резултатите од симулацијата се прикажани на сликата 5.6.1, од каде се заклучува дека SDR детектор со методот на рандомизација има најдобри BER перформанси во однос на другите два разгледувани детектори.

На сликата 5.6.2 е прикажан дополнителен резултат кој покажува како се подобруваат перформансите на SDR детекторот со методот на рандомизација во зависност од бројот на рандомизации L. Може да се забележи значително подобрување на перформансите од $L = 1$ до $L = 50$. Подобрувањето станува помало за

$L > 50$, достигнувајќи граница. Со ова се покажува дека методот на рандомизација обезбедува ефективна апроксимација за семидефинитна релаксација за доволен (но не преголем) број на рандомизации.



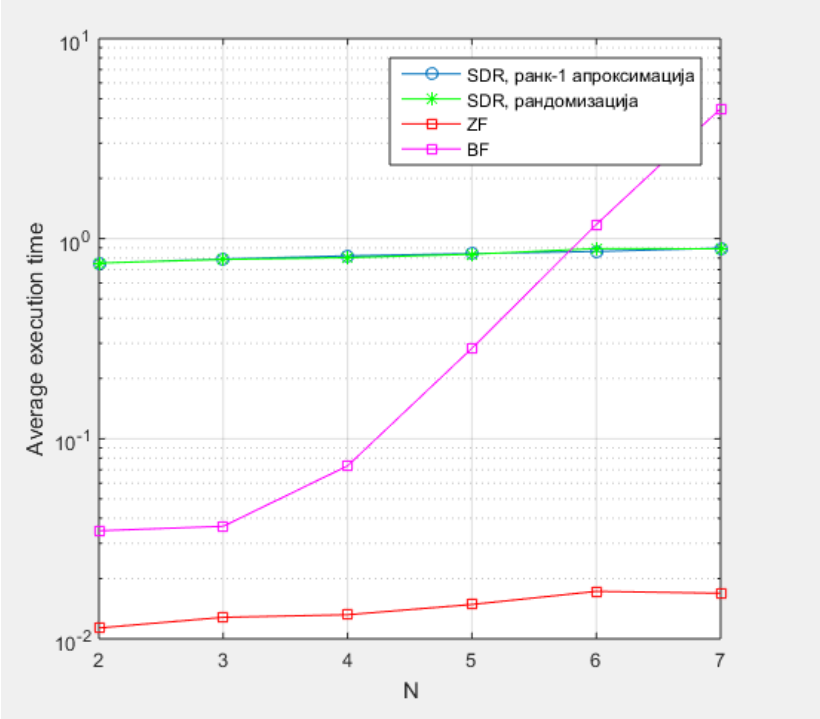
Слика 5.6.1 Перформанси на различни MIMO детектори во QPSK 40x40 MIMO систем



Слика 5.6.2 Перформанси на MIMO детектори со семидефинитна релаксација со различен број на рандомизации

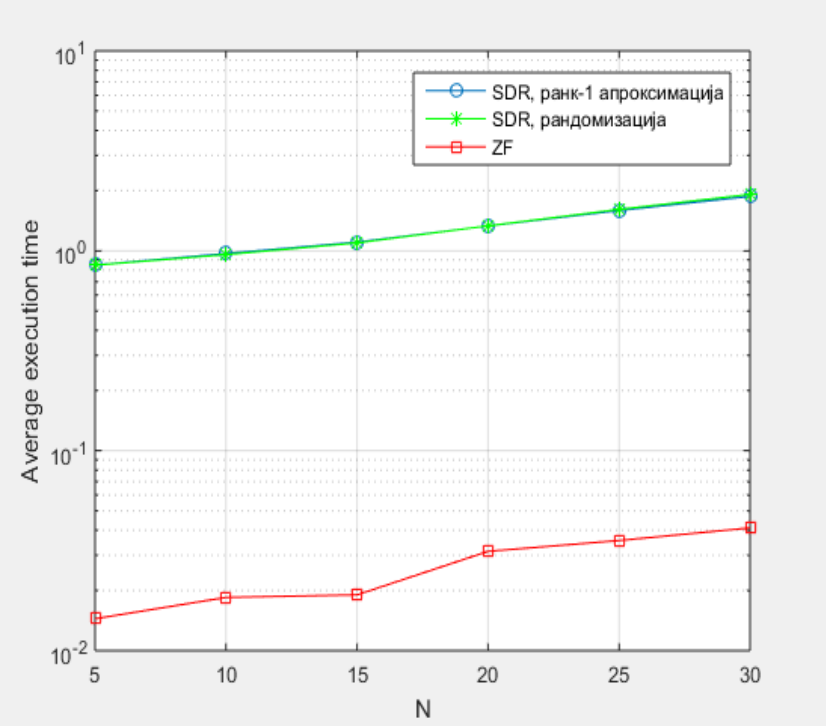
5.6.2 Пресметковна комплексност

За да добиеме целосна слика за поведението на различните MIMO детектори потребно е дополнително да ги споредиме и нивните пресметковни комплексности. Резултатите од овие споредби се прикажани на сликите 5.6.3 и 5.6.4. Симулациите се направени за QPSK модулациска шема за SNR вредност од 12 dB. Се определува просечното време во секунди потребно за истовремена детекција на различен број на испратени симболи N (големина на проблемот). На сликата 5.6.3 интересно е да се нагласи експоненцијално растечката комплексност на решението со „груба сила“ (анг. brute force/BF) наспроти полиномната комплексност на детекторите со семидефинитна релаксација (SDR детектор со методот на апроксимација со сопствени вектори/ранк-1 апроксимација и SDR детектор со методот на рандомизација). Притоа за анализираната големина на проблемот, се забележува дека двата SDR детектори имаат идентична полиномна комплексност независно од користениот метод преку кој се екстрахира решението $\tilde{\boldsymbol{x}}$ на проблемот (5.1.14) од глобалното оптимално решение $\boldsymbol{X}^*$ на конвексниот проблем (5.1.16).



Слика 5.6.3 Споредба на комплексностите на SDR со ранк-1 апроксимација, SDR со рандомизација, ZF и BF MIMO детектори при SNR = 12 dB

Иако побавен од линеарниот zero forcing (ZF) детектор, SDR детекторот е привлечен бидејќи задржува полиномна временска комплексност во однос на големината на проблемот N. Ова може подобро да се забележи на сликата 5.6.4, каде за поголеми вредности на N се дадени само комплексностите на SDR детекторот со ранк-1 апроксимација, SDR детекторот со рандомизација и ZF. Добрите и ветувачки перформанси на SDR MIMO детекторот при детекција на QPSK и бинарни PSK (BPSK) симболи придонеле кон проширување на примената на овој детектор и на други констелации како M - арна PSK и M - арна QAM, [1].



Слика 5.6.4 Споредба на комплексностите на SDR со ранк-1 апроксимација, SDR со рандомизација и ZF MIMO детектори при SNR = 12 dB

6 Заклучок

Главната цел на оваа дипломска работа беше да го анализира методот на семидефинитна релаксација, кој може на лесен и ефикасен начин да се примени во решавањето на неконвексни проблеми од типот на квадратни програми со квадратни ограничувања. Комплетно ја илустиравме идејата за оваа техника, која го запишува оригиналниот неконвексен проблем во форма во која лесно се воочува и отстранува неконвексното ограничување со цел добивање на конвексна семидефинитна програма. Решението на семидефинитната програма може да се добие со алгоритми со внатрешна точка (анг. interior-point) кои имаат полиномна комплексност во најлош случај, па увидовме дека проблемот се сведува на добивање на погодно решение на оригиналниот проблем од оптималното решение на неговата релаксирана форма. Со ова на ефикасен начин се справуваат методот на апроксимација со сопствени вектори и методот на рандомизација, кои во детали ги разгледавме и опишавме.

Можноста ефикасно да се понудат приближно точни решенија на претставници од класата на NP-тешки проблеми наоѓа интересна практична примена во доменот на телекомуникациите и процесирањето на сигнали. Во оваа дипломска работа успешно ја изведовме имплементацијата на детектор со семидефинитна релаксација во MIMO системите, чија задача беше решавање на NP-тешкиот ML детекциски проблем. Двете имплементации на овој детектор – со методот на апроксимација со сопствени вектори (ранк-1 апроксимација) и со методот на рандомизација ги покажаа своите добри перформанси од аспект на помала веројатност на битска грешка при истовремената детекција на симболи. Со цел добивање комплетна слика за поведението на овие детектори беше потребно да го анализираме и нивното просечно време на извршување за различни големини на проблемот, со што ја потврдивме нивната полиномна комплексност.

7 Прилог

Во прилог на дипломската работа се прикажани неколку од кодовите во MATLAB со кои се добиени резултатите од симулациите изложени на крајот од Глава 5.

Код 1. MATLAB скрипта за споредување на BER перформансите на различни MIMO детектори во QPSK 40x40 MIMO систем (Слика 5.6.1).

```
% BER PERFORMANCE %
N = 40;
M = 40;
SNR = 0:3:24;
sigma_snr = sqrt( N*10 .^ ( - (SNR) / 10 ) );
max_no_simulations = 100;
ber_sdr_eigen = zeros(length(SNR),1);
ber_sdr_rand = zeros(length(SNR),1);
ber_zf = zeros(length(SNR),1);
for snr_pt = 1 : length(SNR)
    error_sdr_eigen = 0;
    error_sdr_rand = 0;
    error_zf = 0;
    for simulation_no = 1 : max_no_simulations
        % Генерирање на битите и нивно претворање во QPSK симболи
        s = randi(2,2*N,1)-1;
        s = 2*s - 1;
        s_c = (s(1:length(s)/2) + 1i*s(length(s)/2 + 1:end))/sqrt(2);
        % Генерирање на каналната матрица
        H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));
        % Генерирање на приемниот вектор
        y_c = H_c*s_c + (sigma_snr(snr_pt)/sqrt(2))*(randn(M,1)+1i*randn(M,1));
        % Премин од множеството на комплексни во множеството на реални
        % броеви
        y = sqrt(2)*[real(y_c); imag(y_c)];
        H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];
        % SDR детектор
        [s_sdr_eigen, s_sdr_rand] = sdr_1(y,H);
        % ZF детектор
        s_zf = sign( pinv(H)*y );
        % Пребројување на битските грешки
        error_sdr_eigen = error_sdr_eigen + sum(s_sdr_eigen ~= s);
        error_sdr_rand = error_sdr_rand + sum(s_sdr_rand ~= s);
        error_zf = error_zf + sum(s_zf ~= s);
    end
    % Пресметување на BER во моменталната SNR точка
    ber_sdr_eigen(snr_pt) = error_sdr_eigen / (max_no_simulations*2*N);
    ber_sdr_rand(snr_pt) = error_sdr_rand / (max_no_simulations*2*N);
    ber_zf(snr_pt) = error_zf / (max_no_simulations*2*N);
end
```

```
% Цртање на график
semilogy(SNR,ber_sdr_eigen, '-o');
hold on;
semilogy(SNR,ber_sdr_rand, '-*g');
hold on;
semilogy(SNR,ber_zf, '-sr');
grid on;
legend('SDR, ранк-1 апроксимација','SDR, рандомизација','ZF');
xlabel('SNR (dB)');
ylabel('BER');
```

Код 2. MATLAB скрипта за споредување на BER перформансите на MIMO детектори со семидефинитна релаксација со различен број на рандомизации (Слика 5.6.2).

```
N = 40;
M = 40;
% Број на рандомизации
L =  [1; 5; 20; 50; 80; 120; 160];

SNR = 0:3:24;
sigma_snr = sqrt( N*10 .^ ( - (SNR) / 10 ) );
max_no_simulations = 100;
ber_L = zeros(length(SNR), length(L));
for snr_pt = 1 : length(SNR)
    error_L = zeros(length(L),1);
    for simulation_no = 1 : max_no_simulations
        % Генерирање на битите и нивно претворање во QPSK симболи
        s = randi(2,2*N,1)-1;
        s = 2*s - 1;
        s_c = (s(1:length(s)/2) + 1i*s(length(s)/2 + 1:end))/sqrt(2);
        % Генерирање на каналната матрица
        H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));
        % Генерирање на приемниот вектор
        y_c = H_c*s_c + (sigma_snr(snr_pt)/sqrt(2))*(randn(M,1)+1i*randn(M,1));
        % Премин од C во R
        y = sqrt(2)*[real(y_c); imag(y_c)];
        H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];

        % Детектор
        C = [H'*H, -H'*y; -y'*H, norm(y)^2];
        cvx_begin
        variable X(2*N+1, 2*N+1) symmetric
        minimize(trace(C*X));
        subject to
        for i=1:2*N+1
            X(i,i) == 1;
        end;
        X == semidefinite(2*N+1);
        cvx_end
```

```
        for l_pt = 1 : length(L)
            l = L(l_pt);
            try
                s_sdr_rand_L = rand_L(N,C,X,l);
            catch
                l_pt = l_pt - 1;
                continue;
            end
            error_L(l_pt) = error_L(l_pt) + sum(s_sdr_rand_L ~= s);
        end
    end
    % Пресметување на BER во моменталната SNR точка
    for l_pt = 1 : length(L)
        ber_L(snr_pt, l_pt) = error_L(l_pt) / (max_no_simulations*2*N);
    end
end
```

```
semilogy(SNR,ber_L(:,1), '-o');
hold on;
semilogy(SNR,ber_L(:,2), '-*g');
hold on;
semilogy(SNR,ber_L(:, 3), '-sr');
grid on;
semilogy(SNR,ber_L(:, 4), '-om');
grid on;
semilogy(SNR,ber_L(:, 5), '-*c');
grid on;
semilogy(SNR,ber_L(:, 6), '-sk');
grid on;
semilogy(SNR,ber_L(:, 7), '-oy');
grid on;
legend('L=1','L=5','L=20', 'L=50', 'L=80', 'L=120', 'L=160');
xlabel('SNR (dB)');
ylabel('BER');
```

Код 3. MATLAB скрипта за споредување на комплексностите на SDR со ранк-1 апроксимација, SDR со рандомизација, и ZF MIMO детектори при SNR = 12 dB (Слика 5.6.4.).

```
% AVERAGE EXECUTION TIME %
% Големина на проблемот
N_vrednosti = 5:5:30;
% Однос сигнал-шум
SNR = 12;
% Број на симулации
max_no_simulations = 25;
% Иницијализирање на векторите avg_time
avg_time_sdr_eigen = zeros(length(N_vrednosti),1);
avg_time_sdr_rand = zeros(length(N_vrednosti),1);
avg_time_zf = zeros(length(N_vrednosti),1);
for n = 1:length(N_vrednosti)
    N = N_vrednosti(n);
    sigma_snr = sqrt( N*10 ^ ( - (SNR) / 10 ) );
    M = N;
```

```
    % SDR EIGEN
    tic
    for simulation_no = 1 : max_no_simulations
        u = randi(2,2*N,1)-1;
        u = 2*u - 1;
        s_c = (u(1:2:end) - 1i*u(2:2:end))/sqrt(2);
        H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));
        y_c = H_c*s_c + (sigma_snr/sqrt(2))*(randn(M,1)+1i*randn(M,1));
        y = sqrt(2)*[real(y_c); imag(y_c)];
        H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];
        s_sdr_eigen = sdr_eigen(y,H);
    end
    avg_time_sdr_eigen(n) = toc / max_no_simulations;

    % SDR RAND
    tic
    for simulation_no = 1 : max_no_simulations
        u = randi(2,2*N,1)-1;
        u = 2*u - 1;
        s_c = (u(1:2:end) - 1i*u(2:2:end))/sqrt(2);
        H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));
        y_c = H_c*s_c + (sigma_snr/sqrt(2))*(randn(M,1)+1i*randn(M,1));
        y = sqrt(2)*[real(y_c); imag(y_c)];
        H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];
        s_sdr_rand = sdr_rand(y,H);
    end
    avg_time_sdr_rand(n) = toc/max_no_simulations;


    % ZERO FORCING
    tic
    for simulation_no = 1 : max_no_simulations
        u = randi(2,2*N,1)-1;
        u = 2*u - 1;
        s_c = (u(1:2:end) - 1i*u(2:2:end))/sqrt(2);
        % Generiranje na kanalnata matrica
        H_c = (1/sqrt(2))* (randn(M,N) + 1i*randn(M,N));
        % Generiranje na priemniot vektor
        y_c = H_c*s_c + (sigma_snr/sqrt(2))*(randn(M,1)+1i*randn(M,1));
        % Pretvoranje vo realni vrednosti
        y = sqrt(2)*[real(y_c); imag(y_c)];
        H = [real(H_c) -imag(H_c); imag(H_c) real(H_c)];
        s_zf = zf(y,H);
    end
    avg_time_zf(n) = toc/max_no_simulations;
end
```

8 Референци

[1] Z. -q. Luo, W. -k. Ma, A. M. -c. So, Y. Ye and S. Zhang, "Semidefinite Relaxation of Quadratic Optimization Problems", *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20-34, May 2010, doi: 10.1109/MSP.2010.936019.

[2] d'Aspremont, Alexandre, and Stephen Boyd, "Relaxations and randomized methods for nonconvex QCQPs", *EE392o Class Notes, Stanford University* 1 (2003): 1-16.

[3] "Анализа на алгоритми", Предавање #2, Податочни структури и анализа на алгоритми, Факултет за електротехника и информациски технологии, 2020.

[4] Hong, Daniel, Hyunwoo Lee, and Alex Wei, "Optimal solutions and ranks in the max-cut SDP", *arXiv preprint arXiv:2109.02238* (2021).

[5] Vandenberghe, Lieven, and Stephen Boyd, "Semidefinite programming", *SIAM review* 38.1 (1996): 49-95.

[6] Robert M. Freund, "Symmetric Matrices and Eigendecomposition", Massachusetts Institute of Technology, January 2014.

[7] Herve Abdi, "The Eigen-Decomposition: Eigenvalues and Eigenvectors," Encyclopedia of Measurement and Statistics, 2007.

[8] The Book of Statistical Proofs, "Proof: Expected value of the trace of a matrix", https://statproofbook.github.io/P/mean-tr, Maj 2022.

[9] Cifuentes, Diego, et al. "On the Local Stability of Semidefinite Relaxations." Mathematical Programming, Sept. 2021. Crossref, https://doi.org/10.1007/s10107-021-01696-1.

[10] Ma, Wing-Kin & Ching, Pak-Chung & Ding, Zhi, Semidefinite Relaxation Based Multiuser Detection for M -Ary PSK Multiuser Systems. Signal Processing, IEEE Transactions on. 52. 2862 - 2872. 2004, 10.1109/TSP.2004.834267.

[11] Negrão, João Lucas, Alex Myamoto Mussi, and Taufik Abrão, "Semidefinite relaxation for large scale MIMO detection", *Structure* 1000 (2016): 1.

[12] Jalden, Joakim. *Maximum likelihood detection for the linear MIMO channel*. Diss. 2004.

[13] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[14] Complexity of Algorithms, https://web.mit.edu/urban_or_book/www/book/chapter6/6.2.5.html#:~:text=The%20complexity%20of%20an%20algorithm,division)%20or%20a%20comparison%20between