

*The University of Melbourne*

*Department of Mathematics and Statistics*

# A Look at Composite Concave Programming

Kynan K Marshall

160941

Supervisor: Dr SNIEDOVICH, Moshe

Postgraduate Diploma Thesis

June 2007



# Abstract

We take a general look at optimization and where the topic of Composite Concave Programming fits in and how it differs from classical optimization. As well as looking at what Composite Concave Programming involves, I will have a major focus on Composite Concave Linear programming (CCLP). I will also have a look at quadratic programming/portfolio optimization as it is closely related to the Linear Programming form of the method. In addition, we will solve a CCLP problem using a general Linear Programming package.



# Acknowledgements

Firstly I would like to sincerely thank Dr Moshe Sniedovich for being a fantastic supervisor. Moshe has been incredibly understanding and patient with me and always very helpful when it was needed.

To my fellow honours classmates and friends, you have all been wonderful. Thank you for always taking the time out to help me and answer questions when needed, this goes out to last years honours participants as well as this years. Thank you also to my family for all the support I have been given over my many years of schooling.



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Review of Literature</b>	<b>5</b>
2.1	Composite Concave Programming . . . . .	5
2.2	Composite Concave Linear Programming . . . . .	6
2.3	Soft Constraints . . . . .	9
2.4	Portfolio Optimization . . . . .	10
<b>3</b>	<b>Composite Concave Programming</b>	<b>13</b>
3.1	Concave Programming . . . . .	13
3.2	Composite Functions . . . . .	18
3.2.1	Ratio Functions . . . . .	21
3.2.2	Multiplicative Functions . . . . .	21
3.2.3	Additive Functions . . . . .	22
<b>4</b>	<b>Composite Concave Linear Programming</b>	<b>23</b>
4.1	Formulation . . . . .	23
4.2	Optimality Conditions . . . . .	24
4.3	Search Space . . . . .	27
4.4	Algorithms . . . . .	30
4.4.1	Mid-Point Algorithm . . . . .	30
<b>5</b>	<b>Numerical Example</b>	<b>33</b>
<b>6</b>	<b>Conclusion/Discussion</b>	<b>41</b>
<b>7</b>	<b>Appendix</b>	<b>43</b>
7.1	Appendix A . . . . .	43
7.2	Appendix B . . . . .	44
7.3	Appendix C . . . . .	47



# Chapter 1

## Introduction

The main aim of this thesis is to examine the method of Composite Concave Programming (CCP/c-programming), and in particular the sub class of Composite Concave Linear Programming (CCLP).

We explore the optimization of Composite Concave functions, along the way introducing this method which differs from many other optimization method as we are searching for the global optimal to these problems. This is done with the transformation of these problems to make them linear, thus allowing us to solve them with our beloved Linear Programming packages.

Along the way, we will be exploring the make up of the problems, exploring composite functions, concave functions and exploiting these properties to make it possible to solve for a global optimal.

There is also a look at uses for c-programming, ranging from portfolio optimization to use with soft constraints.

### **Chapter 2: Review of Literature**

An overview of some of the literature of the topic.

### **Chapter 3: Composite Concave Programming**

Spells out what CCP is with relation to Concave programming and Composite Functions.

### **Chapter 4: Composite Concave Linear Programming**

Concentrates on CCLP's, including the formulation, optimality conditions, search space and also algorithms.

## **Chapter 5: Numerical Example**

Solve a numerical example of a CCLP, with the use of a Linear Programming package.

## **Chapter 6: Conclusion/Discusion**

# Chapter 2

## Review of Literature

In this chapter we take a general over view of papers written on the topics of Composite Concave Programming and Composite Concave Linear Programing. These explore the use of the method, the problems they can be applied to and some of the most important sufficient condition needed. This will give an over view of the topic and a taste for it before going into it deeper in the ensuing chapters.

### 2.1 Composite Concave Programming

The class of non-linear optimization problems under consideration are called c-programming (composite concave programming). Sniedovich [1] gives the generic problem as follows:

$$\begin{aligned} & \textit{Problem } T \\ & \tau := \min_{z \in Z} r(z) := \phi(u(z)) \end{aligned} \tag{2.1}$$

where  $u$  is a function on some set  $Z$  with values in  $R^k$ , and  $\phi$  is a real-valued function on  $u(Z) := \{u(z) : z \in Z\}$ . Let  $Z^*$  denote the set of optimal solutions to this problem. We assume that  $Z^*$  is not empty.

The tactic that composite concave programming employs to solve problems of this type is to tackle them indirectly. Instead of solving the *Problem T*, it obtains its solution by solving a linearization of  $\phi$  with respect to  $u$ . That is associated with

(2.1) we consider the parametric problem:

*Problem T*( $\lambda$ )

$$t(\lambda) := \min_{z \in Z} r_\lambda(z) \quad (2.2)$$

$$:= \lambda^T u(z) \quad (2.3)$$

$$:= \sum_{m=1}^k \lambda_m u_m(z), \quad \lambda \in R^k \quad (2.4)$$

where  $u_m(z)$  and  $\lambda_m$  denotes the  $m$ -th component of  $u(z)$  and  $\lambda$  respectively, and  $\lambda^T$  denotes the transpose of  $\lambda$ . Let  $Z^*(\lambda)$  denote the set of optimal solutions to *Problem T*( $\lambda$ ). It is assumed that for each  $\lambda \in R^k$  the set  $Z^*(\lambda)$  consists of at least one element.

The reasoning in seeking the solution of *Problem T* through the use of *Problem T*( $\lambda$ ) is that in many cases the original problem defies standard optimization methods, where as the new problem is susceptible to these methods.

The theory of c-programming examines the relationship between *Problem T* to *Problem T*( $\lambda$ ) and the conditions that have to be met to use this method, i.e. Differentiability, pseudoconcavity and etc.

## 2.2 Composite Concave Linear Programming

Sniedovich, Emmanuel and Findley [4], examines how the simplex method can be used as a global optimizer, in other words, as a method for getting global optimal solutions to nonlinear programming problems that may have more than one local optimal and where there is no guarantee that a local optimal is a global one. These are problems whose objective functions are not psedoconvex (assuming opt is min). The difficulty with these problems is that when the problem is not psedoconvex, it is often difficult to check whether a given optima is a global one or not. This paper discusses how this difficulty can be over come for a certain class of problems by using the technique of c-programming. This paper examines the expressive power of c-programming and the role that it can play in global optimization when used with the simplex method. i.e, using the simplex method for problems other than linear programming problems which it has been doing for years. The following

composite concave problems are examined:

*Problem P*

$$\tau := \min_{x \in X} g(x) := \phi(Cx + d) \quad (2.5)$$

where

$$X := \{x \in \mathfrak{R}^n : Ax = b, x \geq 0\}, \quad (2.6)$$

$\mathfrak{R}$  denotes the real line,  $A$  is an  $m \times n$  matrix,  $b$  is an  $m - dimensional$  vector,  $C$  is a  $k \times n$  matrix,  $d$  is a  $k - dimensional$  vector and  $\phi$  is a real valued function on  $\mathfrak{R}^k$ . Let  $X^*$  denote the set of (global) optimal solutions to *Problem P*. Now let us take a look at some examples.

**Example 2.1** For the function defined by

$$g(x) = (3x_1 + 6x_2 + 1)(5x_1 + 2x_2 + 3), x \in \mathfrak{R}^2 \quad (2.7)$$

we can set  $k = 2$ ,  $d = (1, 3)$ ,  $C = \begin{bmatrix} 3 & 6 \\ 5 & 2 \end{bmatrix}$  and

$$\phi(v, w) = v \times w, v, w \in \mathfrak{R}, \quad (2.8)$$

where as for the function defined by

$$g(x) = 3x_1 + 6x_2 + 1 - (5x_1 + 2x_2 + 3)^2, x \in \mathfrak{R}^2 \quad (2.9)$$

$k$ ,  $d$ , and  $C$  can be set as above, and  $\phi$  defined as follows:

$$\phi(u, w) = u - w^2, v, w \in \mathfrak{R}. \quad (2.10)$$

It is important to note that in both cases the function  $g$  is not pseudoconvex with respect to  $x$ . So the difficulty is that any local optimal found is not necessarily a global one. This is where the c-programming approach comes into play.

We are now going to have a look at a two-dimensional linear c-programming problems, this is supposing that  $k = 2$ . Then Problem P can be written as follows:

*Problem P2*

$$\tau := \min_{x \in X} g(x) := \phi(C_1x + d_1, C_2x + d_2) \quad (2.11)$$

and the parametric problem of c-programming has the following form:

$$\tau(\lambda) := \min_{x \in X} g(x; \lambda) := \lambda_1 C_1x + \lambda_2 C_2x, \lambda \in \mathfrak{R}^2. \quad (2.12)$$

Ignoring the instance where  $\lambda_1 = 0$ , and dividing the right-hand side by  $|\lambda_1|$ , the following parametric problem is obtained:

*Problem P2*( $\beta$ )

$$\tau(\beta) := \min_{x \in X} g(x; \beta) := sC_1x + \beta C_2x, \beta \in \Re, s \in \{1, 0, -1\} \quad (2.13)$$

$$= (sC_1 + \beta C_2)x \quad (2.14)$$

where  $\beta = \frac{\lambda_2}{|\lambda_1|}$ . The scalar  $s$  is not regarded as a parameter, instead it is treated as an indicator that sometimes it may be necessary to multiply  $C_1$  by -1 and in other instances by 0.

The significance of the form given in (2.12)-(2.13) is that it is receptive to the conventional parametric analysis of the simplex method. That is solving *Problem P2*( $\beta$ ) for a range of values for  $\beta$  is not much more costly than solving a single linear programming problem of the same size.

The simplex algorithm used for the problem consists of three steps.

1. It may be necessary to determine the interval, call it  $B$ , in which  $\beta$  is varied. This is usually done by inspection. But if no obvious lower and upper bound can be found for  $\beta$ , then  $B = [-M, M]$ , where  $M$  is an adequately large number. In many cases the parametric simplex procedure itself will take care of this matter.
2. Using conventional parametric linear programming techniques, *Problem P2*( $\beta$ ) is solved for a finite sequence of values of  $\beta \in B$ , say  $\{\beta^{(i)}\}$ . Let  $x(\beta^{(i)})$  denote the optimal solution recovered for *Problem P2*( $\beta^{(i)}$ ). The parametric analysis techniques guarantee that for any  $\beta \in B$  there is some  $\beta^{(i)}$  such that  $x(\beta^{(i)})$  is an optimal solution for *Problem P2*( $\beta$ )
3. The optimal solution to *Problem P2*( $\beta$ ) is recovered by selecting an  $i^*$  such that  $x(\beta^{(i^*)})$  minimizes  $g(x(\beta^{(i)}))$  over  $\{x(\beta^{(i)})\}$ .

So from the above, it follows that it shows  $k = 2$  and the composite function  $\phi$  is differentiable and pseudoconcave, then *Problem P* can be solved efficiently by conventional linear programming techniques. Extensive experiments conducted with algorithms based on the analysis are reported on in Macalalag and Sniedovich [7].

C-programming can also be applied in situation where  $k > 2$ . The basic format is similar except that the parameter  $\lambda$  relating *Problem P* and *Problem P* ( $\lambda$ ) is a  $k$ -vector and the conditions of differentiability and pseudoconcavity takes place in some convex set  $U \subseteq \Re^k$ .

This paper also concerns itself with the types of composite functions which could be solved by the previously discussed two-dimensional linear c-programming. The ones looked at are Ratio, Multiplicative and Additive functions of the form:

*Ratio :*

$$\phi(v, w) := \frac{v}{w}, \quad v \in \mathfrak{R}, w \in \mathfrak{R}^+ \quad (2.15)$$

*Multiplicative :*

$$\phi(v, w) := \sigma(v)\varphi(w) \quad (2.16)$$

*Additive :*

$$\phi(v, w) := \sigma(v) + \varphi(w), \quad v, w \in \mathfrak{R} \quad (2.17)$$

We will be having a more detailed look later.

## 2.3 Soft Constraints

Byrne, Sniedovich and Churilov [3] present a unconventional approach to problems involving soft constraints based on composite concave programming. They examine a couple of methods for handling problems with soft constraints, including RHS parametric analysis, goal programming and the one we are interested in, C-programming.

The problem under consideration is as follows:

*Problem P(r)*

$$p(r) := \max_{x \in H} f(x) \text{ subject to } s(x) = r, \quad r \in R := [\underline{r}, \bar{r}] \quad (2.18)$$

where  $[a, b]$  denote the convex hull of  $\{a, b\}$ , and  $\underline{r}$  and  $\bar{r}$  are lower and upper bounds for the value of  $r$  in the right hand side of the constraint. In the c-programming model,  $s(x)$  is viewed as vector of  $k$  'resources' that must be purchased at a price specified by some given real-valued cost function,  $\varphi$ . So the c-programming model for handling *Problem Soft* calls for the solution of the following:

*Problem C*

$$c := \max_{x \in H} \{t(x) := f(x) - \varphi(s(x))\} \quad (2.19)$$

where  $\varphi$  is some real-valued function on  $s(H) := \{s(x) : x \in H\}$ . Let  $H^c$  denote the set of optimal solutions of this problem. With the inclusion of the cost function term  $\varphi(s(x))$ , this may now make the *Problem C* difficult to solve. Since we want

to solve *Problem C* 'efficiently', this is exactly where c-programming comes in. Now *Problem C* can be solved by the following parametric problem:

$$\begin{aligned} & \textit{Problem C}(\lambda) \\ & c(\lambda) := \max_{x \in X} \{f(x) - \lambda s(x)\}, \quad \lambda \in \mathfrak{R}^k \end{aligned} \quad (2.20)$$

where  $\lambda$  is viewed as a row vector and  $\lambda s(x) := \sum_{i=1}^k \lambda_i s_i(x)$ . Let  $H^c(\lambda)$  denote the set of optimal solutions of *Problem C*( $\lambda$ ).

**Theorem 2.2** *Assume that  $\varphi$  is differentiable and concave with respect to  $s$  on some open convex set  $U$  such that  $s(H) \subseteq U$ . Then,*

$$x \in H^c(\nabla\varphi(s(x))), \quad \forall x \in H^c \quad (2.21)$$

and furthermore,

$$H^c(\nabla\varphi(s(x))) \subseteq H^c, \quad \forall x \in H^c \quad (2.22)$$

where  $\nabla\varphi(s(x))$  denotes the gradient of  $\varphi$  with respect to  $s$  at  $s(x)$ .

The theorem guarantees that, subject to the differentiability and convexity conditions, there exists a  $\lambda^o$  such that any optimal solution of *Problem C*( $\lambda^o$ ) is also an optimal solution of *Problem C*.

## 2.4 Portfolio Optimization

Churilov, Bomze, Sniedovich and Ralph [2], illustrate how composite concave programming is used to perform Hyper Sensitivity Analysis (HSA) in the area of Portfolio Optimization Problems. This problem is a general mean-variance optimization problem involving  $n$  securities that are jointly distributed with a mean vector  $\mu^T = (\mu_1, \dots, \mu_n)$  and covariance matrix  $\frac{1}{2}C = (\sigma_{ij})$ . An investor has to select a *portfolio*, a vector  $x$  of holdings with separate returns represented by vector  $r^T = (r_1, \dots, r_n)$ . The expected return and variance of the portfolio is given by  $E(x) := \mu^T x$  and  $V(x) := \frac{1}{2}x^T C x$ . The portfolio is chosen subject to linear constraints  $Ax = b$ ,  $x \geq 0$ , where  $A$  is a  $(m \times n)$  matrix, and  $b$  is a vector of  $m$  components. A pair  $(E(x), V(x))$  is said to be *feasible* if the corresponding portfolio  $x$  is feasible, i.e. if it satisfies the above-mentioned constraints.

This classical formulation of general mean-variance optimization problems relies on the following notion of efficiency.

**Definition 2.3** [8] *A feasible pair  $(E(x'), V(x'))$  is inefficient if there is another feasible pair  $(E(x''), V(x''))$ , such that either:*

1.  $E(x'') > E(x')$  and  $V(x'') \leq V(x')$ ; or
2.  $V(x'') < V(x')$  and  $E(x'') \geq E(x')$ .

A feasible pair  $(E(x), V(x))$  is efficient if it is not inefficient.

Matrix  $C$  is covariance matrix, and therefore is positive semi-definite. If  $C$  is strictly positive definite, then for a given efficient pair  $(E(x), V(x))$ , there exists a unique corresponding efficient portfolio.

**Definition 2.4** [8] A set of efficient portfolios is said to be complete and non-redundant if it contains one and only one efficient portfolio for each efficient pair  $(E(x), V(x))$ .

This paper also gives use our first look at a version of what will become known as the *fundamental theorem of CCLP*.

**Theorem 2.5** [9]

Let function  $g$  be differentiable and pseudoconvex with respect to  $u(x) = (a(x), v(x))$  on some open convex set containing  $\{u(x) : x \in X\}$ . Then, any optimal solution for Problem  $P$  is also optimal with respect to the parametric problem it generates, namely

$$x \in X^* \Rightarrow x \in X^*(\nabla g(u(x))). \quad (2.23)$$

Moreover, any optimal solution to this parametric problem is also optimal with respect to Problem  $P$ , namely

$$x \in X^* \Rightarrow X^*(\nabla g(u(x))) \subseteq X^*. \quad (2.24)$$

Theorem, 2.5, guarantees that there exists an optimal value of the parameter



# Chapter 3

## Composite Concave Programming

Now we will have a look at composite programming in general and also composite functions and how combining these two, makes it possible for us to solve these particular problems.

### 3.1 Concave Programming

It is now time to have a look at the basics of Concave Programming and what it has to do with the relationship between the following two problems:

$$\begin{array}{ll} \textit{Problem } P & \textit{Problem } P(\lambda) \\ z^* := \min_{x \in X} f(x) & z^*(\lambda) := \min_{x \in X} \lambda^T x \end{array} \quad (3.1)$$

where

$$X \subseteq \Re^n.$$

$f$  is a real valued function on  $X$ .

$$\lambda \in \Re^n.$$

In words, *Problem*  $P(\lambda)$  is obtained from *Problem*  $P$  by linearizing the objective function,  $f$ , of *Problem*  $P$  where  $\lambda$  is a *parameter* regarded as a constant that is used in the linearization. *Problem*  $P$  is the *target problem*, i.e. the problem we have to solve, and *Problem*  $P(\lambda)$  is the *parametric problem*. Let  $X^*$  denote the

set of optimal solutions to *Problem P* and let  $X^*(\lambda)$  denote the set of optimal solutions of *Problem P*( $\lambda$ ),  $\lambda \in \mathfrak{R}^n$ .

**Example 3.1** Consider the following problems and their linear equivalents:

$$z^* := \min_{x \in X} \sqrt{x} \Rightarrow z^*(\lambda) := \min_{x \in X} \lambda x \quad (3.2)$$

$$z^* := \min_{x_1 \in X} \log x_2 - x^2 \Rightarrow z^*(\lambda) := \min_{x \in X} \lambda_1 x_1 - \lambda_2 x_1 \quad (3.3)$$

$$z^* := \min_{x \in X} \sqrt{x_1} + \sqrt{x_2} + \sqrt{x_3} \Rightarrow z^*(\lambda) := \min_{x \in X} \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \quad (3.4)$$

where  $\lambda \in \Lambda$  - some subset of  $\mathfrak{R}^k$ .

It has to be pointed out that we are looking for *global* and not just *local* optimal solutions, thus  $x^* \in X$  is an optimal solution to *Problem P* if and only if  $f(x^*) \leq f(x), \forall x \in X$  (minimization). Since the objective function of *Problem P*( $\lambda$ ) is linear with  $x$ , it is often easier to solve *Problem P*( $\lambda$ ) for any given value of  $\lambda$  than it is to solve *Problem P*, but this is not always the case. Now this leads to the following question; *Is there a  $\lambda^* \in \mathfrak{R}^n$  such that any optimal solution generated for *Problem P*( $\lambda^*$ ) is also optimal for *Problem P*?*

$\lambda^*$  is referred to as the optimal  $\lambda$ . More formally, we say that  $\lambda^* \in \mathfrak{R}^n$  is optimal if and only if  $X^*(\lambda^*) \subseteq X^*$ . The question is this; *does an optimal  $\lambda$  exist?*

The answer to this question can be answered in two parts:

- an optimal  $\lambda$  does not always exist.
- but there are simple sufficient conditions for the existence of an optimal  $\lambda$

These sufficient conditions will be stated shortly.

To shed some light on the situation, consider the following problem:

$$z^* := \min_{-2 \leq x \leq 2} \frac{x^2}{2} + 2 \quad (3.5)$$

We can set  $X = [-2, 2]$  and  $f(x) = \frac{x^2}{2} + 2$ . It is easy to show that in this case  $X^* = \{0\}$  and

$$X^*(\lambda) = \begin{cases} \{-2\} & , \lambda > 0 \\ X & , \lambda = 0 \\ \{2\} & , \lambda < 0 \end{cases} \quad (3.6)$$

So in this case there is no optimal  $\lambda$  even though  $X^*(0)$  contains an optimal solution to *Problem P*.

Now lets, look at the following problem:

$$z^* := \min_{-2 \leq x \leq 2} \frac{-x^2}{2} + 2 \quad (3.7)$$

again can set  $X = [-2, 2]$  and  $f(x) = \frac{-x^2}{2} + 2$ . Here we have  $X^* = \{-2, 2\}$  and  $X^*(\lambda)$  as is in (3.6). Hence in this case there is infinitely many optimal  $\lambda$ 's: in-fact all  $\lambda \in \mathfrak{R} \setminus \{0\}$  are optimal.

We now examine some simple sufficient conditions for the existence of optimal  $\lambda$ 's.

**Theorem 3.2** [6] *Assume that  $f$  is differentiable and concave on some open convex set containing  $X$  and that  $X^*$  is not empty. Then there is a  $\lambda^* \in \mathfrak{R}^n$  such that  $X^*(\lambda^*) \subseteq X^*$ .*

This means that if *Problem P* possesses an optimal solution and its objective function  $f$  is differentiable and concave then an optimal  $\lambda$  exists.

We now have a guarantee for the existence of a optimal  $\lambda$  under certain conditions. We now examine how to determine such values.

**Theorem 3.3** [6] *Assume that in the framework of (3.1) the objective function  $f$  is differentiable and concave on some open convex set containing  $X$  and that  $X^*$  is not empty. Let  $x^*$  be any element of  $X^*$ . Then  $\lambda^* = \nabla f(x^*)$  is an optimal  $\lambda$ . In short,*

$$X^*(\nabla f(x)) \subseteq X^*, \forall x \in X^*. \quad (3.8)$$

*Reminder:*  $\nabla f(x)$  denotes the gradient of  $f$  at  $x$ .

In words, the gradient of  $f$  at any point that is a optimal solution to *Problem P* is an optimal  $\lambda$ , subject to the sufficient conditions (theorem 3.2). Using the information obtained from  $\nabla f(x)$ , we can now set up a *search space* for an optimal  $\lambda$ . With this space, let  $\Lambda$  be a subset of  $\mathfrak{R}^n$  such that

$$\{\nabla f(x) : x \in X\} \subseteq \Lambda \quad (3.9)$$

Since this set contains all of the optimal  $\lambda$ 's implied by Theorem 3.3, we now only have to solve *Problem P*( $\lambda$ ) for all  $\lambda \in \Lambda$  to obtain our optimal solution to *Problem P* and thus identifying the solution that minimizes  $f$ . More formally, let  $x^*(\lambda)$  represent the optimal solution found for *Problem P*( $\lambda$ ). Then the plan is to solve

$$\lambda^o := \arg \min_{\lambda \in \Lambda} f(x^*(\lambda)) \quad (3.10)$$

observing that subject to Theorem 3.2,  $x^*(\lambda^o)$  would be an optimal solution to the *Problem P*.

It is quite easy to formulate the search for optimal  $\lambda$ 's into a pseudo-operational plan, as expressed below:

Step 1: Determine the appropriate search space  $\Lambda \subset \mathfrak{R}^n$ .

Step 2: Set  $z^o = \infty$

Step 3: For each  $\lambda \in \Lambda$  (according to some order) Do:

Set  $x^*(\lambda) = \arg \min_{x \in X} \lambda^T x$

if  $f(x^*(\lambda)) < z^o$  set  $z^o = f(x^*(\lambda))$ ,  $\lambda^o = \lambda$  and  $x^o = x^*(\lambda)$

End Do

And of course this is subject to our sufficient conditions. Upon completion this procedure will generate an optimal solution to *Problem P*( $\lambda$ ), and also an optimal solution  $x^o$  for *Problem P*.

But there is a problem with this formulation, it requires us to solve *Problem P*( $\lambda$ ) for every  $\lambda$  in  $\Lambda$ . Even if *Problem P*( $\lambda$ ) is very simple to solve for any given  $\lambda$ ,  $\Lambda$  could be very large and if this is the case the above plan will not be feasible. And give that  $\Lambda$  essentially consists of  $\nabla f(x)$  for all  $x \in X$ , many applications will have infinitely many components, in which case our plan would be in trouble.

So how can this problem be overcome?

Firstly, it is important to note that the problem is caused by the fact that the search space for optimal  $\lambda$  is of high dimension, that is  $\lambda = (\lambda_1, \dots, \lambda_n)$  where  $n$  is the dimension of the decision vector  $x \in \mathfrak{R}^n$ .

One possible solution would be to restrict the value of  $n$ , i.e. require  $n$  to be small, say no larger than 5 or 6. But this is not a solution at all, as it means that we restrict ourselves to problems with only 5 or 6 decision variables, so this is out of the question.

How can we keep  $n$  large yet reduce the dimension of the search space? There is a simply remedy for this: let the objective function  $f$  of *Problem P* depend only on  $k \leq n$  of the  $n$  decision variables rather than all of them. This makes it possible to have  $n$  as large as we want and  $k$  as small as we want at the same time. With this remedy,  $\lambda$  is now a vector of the smaller  $\mathfrak{R}^k$  rather than  $\mathfrak{R}^n$ . We only need  $k \leq n$  but we are only going to be interested in cases where  $k$  is very small, no greater than say 4 or 5.

The task now is not only to come up with a reformulation of the *Problem P* for which  $k$  is small. We also want a formulation for which  $k$  is small and at the same

time the corresponding parametric problem, *Problem P*( $\lambda$ ), is relatively easy. The following example illustrates such a case.

**Example 3.4** *Consider the nonlinear optimization problem:*

$$z^* := \sqrt{\sum_{i=1}^{10000} d_i x_i} + \log \left( \sum_{i=1}^{10000} c_i x_i \right), \quad X \subset \mathfrak{R}^{10000} \quad (3.11)$$

in which case  $n = 10000$ , and formally

$$f(x) = \min_{x \in X} \sqrt{\sum_{i=1}^{10000} d_i x_i} + \log \left( \sum_{i=1}^{10000} c_i x_i \right), \quad x \in X \subset \mathfrak{R}^{10000} \quad (3.12)$$

Note that  $f$  depends on each one of the 10000 decision variables. But in this case it is possible to reformulate the objective function so it does not depend on all of the decision variables:

$$z^* := \min_{x,u} \sqrt{u_1} + \log(u_2), \quad X \subset \mathfrak{R}^{10000} \quad (3.13)$$

$$x \in X$$

$$u_1 = \sum_{i=1}^{10000} d_i x_i \quad (3.14)$$

$$u_2 = \sum_{i=1}^{10000} c_i x_i \quad (3.15)$$

Here there are 10002 decision variables but the objective function depends on only  $k = 2$  of them.

These two representations are equivalent in that there is a one to one correspondence between their feasible and optimal solutions, and their objective functions are also equivalent.

But there is no guarantee that for any given nonlinear optimization problem there is a possible formation for which an equivalent problem whose objective function depends only on a small number of decision variables and also has a parametric problem that is easy to solve. This is just a method for a class of problems for which it is possible to form an equivalent problem that depends on only a small number of decision variables. Of course there are many cases where this is not possible. For example, consider the following

### Example 3.5

$$z^* := \min_{x \in X} \sum_{i=1}^{30} \sqrt{x_i}, \quad X \subset \mathfrak{R}^{30} \quad (3.16)$$

$$x_i \geq 1, \quad i = 1, \dots, 30 \quad (3.17)$$

It looks as if that the objective function depends on all of the decision variables and there is not a great deal can do about it.

Though this may not be a solution to all problems, there are many that are very acceptable to this method. That is we can obtain a small  $k$  even if the problem has many decision variables.

## 3.2 Composite Functions

We now examine composite functions, and how they are used in composite concave programming. Even though composite functions play an important role in mathematics, i.e. as the framework in the *Chain Rule* for differentiation, they are not used in the framework for classification of *optimization problems*.

For the purpose of our look at CCLP, and more broadly at CCP, we will be considering the case where the objective function  $f$  of *Problem P* has the following representation:

$$f(x) = g(u(x)), \quad x \in X \subseteq \mathfrak{R}^n \quad (3.18)$$

where

$u$  is a function on  $X$  with values in  $\mathfrak{R}^k$ ,  $k \leq n$  such that

$$u(x) = (u_1(x), \dots, u_k(x))$$

and  $u_j$  is a real valued function on  $X$ ,  $j = 1, \dots, k$ .

$g$  is a real valued function on  $u(X) := \{u(x) : x \in X\}$ .

We shall refer to  $u$  as the *inner function* and to  $g$  as the *outer function*.

It has to be pointed out that there is always such a representation for such functions, but at the same time there could be numerous different representations for the same function.

For example, consider the function  $f$  defined by

$$f(x) = \frac{(4x_1 + 3x_2)^2}{\sqrt{(1 + 2x_3^4)}}, \quad x \in X \subset \mathfrak{R}^3 \quad (3.19)$$

where  $X$  is defined by a system of linear constraints.

What follows is three decomposition schemes for  $f$ , and the question arises: which one is better, or more suitable, for our purpose?

Scheme 1	Scheme 2	Scheme 3
$k = 2$ $u_1 = (4x_1 + 3x_2)^2$ $u_2 = 1 + 2x_3^4$ $g(u(x)) = \frac{u_1(x)}{\sqrt{u_2(x)}}$	$k = 2$ $u_1(x) = 4x_1 + 3x_2$ $u_2 = \sqrt{1 + 2x_3^4}$ $g(u(x)) = \frac{u_1^2(x)}{u_2(x)}$	$k = 2$ $u_1 = 4x_1 + 3x_2$ $u_2 = x_3$ $g(u(x)) = \frac{u_1^2(x)}{\sqrt{1+2u_3^4(x)}}$

Table 3.1:

It should be noted that there are other valid schemes for equation 3.19, and perhaps better schemes for our situation. But we are just going to look at the ones in the above table and see which is best/better for our use in CCLP. The choice of these composition schemes may have an effect on how easy/difficult it will be to solve *Problem P*( $\lambda$ ).

So it is important to understand how the choice of a composition scheme for  $f$  affects the structure of *Problem P*( $\lambda$ ). In the composition scheme of  $f(x) = g(u(x))$ , the parametric problem is obtained by linearizing function  $g$  rather than function  $f$  and that the linearization is with respect to  $u(x)$  rather than  $x$ . This leads to the process of generating *Problem P*( $\lambda$ ) for a given composition scheme can be described as follows:

*Problem P* :

$$\min_{x \in X} f(x) \Rightarrow \min_{x \in X} g(u(x)) \Rightarrow \min_{x \in X, y=u(x)} g(y) \quad (3.20)$$

*Problem P*( $\lambda$ ) :

$$\min_{x \in X} \lambda^T x \Rightarrow \min_{x \in X} \lambda^T u(x) \Rightarrow \min_{x \in X, y=u(x)} \lambda^T y \quad (3.21)$$

Note that the framework of *Problem P*( $\lambda$ ), the parameter vector  $\lambda$  changes its dimension from  $n$  to  $k$  and that there are two (equivalent) choices regarding where  $u(x)$  appears in the formulation:

- In the objective function:  $\min_{x \in X} \lambda^T u(x)$ .
- In the constraints:  $\min_{x \in X, y=u(x)} \lambda^T y$ .

The lesson in this is that if  $u(x)$  is nonlinear, then the parametric problem induced by the composition scheme is also not linear.

Now to illustrate how this method works and the result of the choices made in regard to the composition scheme. This will be done on the three schemes from above and compare the parametric problems they lead to.

Scheme 1 results in the following setup:

*Problem P :*

$$\frac{(4x_1 + 3x_2)^2}{\sqrt{(1 + 2x_3^4)}} \Rightarrow \min_{x \in X} \frac{u_1(x)}{\sqrt{u_2(x)}} \Rightarrow \min_{x \in X, y_1 = (4x_1 + 3x_2)^2, y_2 = 1 + 2x_3^4} \frac{y_1}{\sqrt{y_2}} \quad (3.22)$$

*Problem P(λ) :*

$$\min_{x \in X} \lambda^T x \Rightarrow \min_{x \in X} \lambda^T u(x) \Rightarrow \min_{x \in X, y_1 = (4x_1 + 3x_2)^2, y_2 = 1 + 2x_3^4} \lambda^T y \quad (3.23)$$

Note that the resultant parametric problem is not linear.

Scheme 2 results in the following setup:

*Problem P :*

$$\frac{(4x_1 + 3x_2)^2}{\sqrt{(1 + 2x_3^4)}} \Rightarrow \min_{x \in X} \frac{u_1^2(x)}{u_2} \Rightarrow \min_{x \in X, y_1 = 4x_1 + 3x_2, y_2 = \sqrt{1 + 2x_3^4}} \frac{y_1^2}{y_2} \quad (3.24)$$

*Problem P(λ) :*

$$\min_{x \in X} \lambda^T x \Rightarrow \min_{x \in X} \lambda^T u(x) \Rightarrow \min_{x \in X, y_1 = 4x_1 + 3x_2, y_2 = \sqrt{1 + 2x_3^4}} \lambda^T y \quad (3.25)$$

Note that the resultant parametric problem is not linear.

Scheme 3 results in the following setup:

*Problem P :*

$$\frac{(4x_1 + 3x_2)^2}{\sqrt{(1 + 2x_3^4)}} \Rightarrow \min_{x \in X} \frac{u_1^2(x)}{\sqrt{1 + 2u_2^4(x)}} \Rightarrow \min_{x \in X, y_1 = 4x_1 + 3x_2, y_2 = x_3} \frac{y_1^2}{\sqrt{1 + 2y_2^4}} \quad (3.26)$$

*Problem P(λ) :*

$$\min_{x \in X} \lambda^T x \Rightarrow \min_{x \in X} \lambda^T u(x) \Rightarrow \min_{x \in X, y_1 = 4x_1 + 3x_2, y_2 = x_3} \lambda^T y \quad (3.27)$$

Note that the resultant parametric problem is linear, and recalling that  $X \subset \mathfrak{R}^3$  is defined by a system of linear equations. And since we are intending to solve these forms of problems with the aid of a LP package, Scheme 3 appears to be much better than the other two.

#### GENERIC CLASSES:

It is all so important to examine the types of composite functions that readily lend themselves to the two-dimensional linear c-programming ( $k = 2$ ). We will focus on three major classes of problems.

### 3.2.1 Ratio Functions

The function  $f(x)$  defined by

$$f(v, w) := \frac{v}{w}, \quad v \in \mathfrak{R}, w \in \mathfrak{R}^+ := \{r \in \mathfrak{R} : r > 0\} \quad (3.28)$$

is differentiable and pseudolinear on  $\mathfrak{R} \times \mathfrak{R}^+$ .

For example, it covers composite objective functions of the form

$$f(v, w) := \frac{g(v)}{w}, \quad v, w \in \mathfrak{R}, w > 0 \quad (3.29)$$

where  $g(v)$  is a differentiable concave function, which leads to  $f(v, w)$  being pseudoconcave with respect to  $(v, w)$ . The same situation occurs in cases where

$$f(v, w) := \frac{v}{h(w)}, \quad v, w \in \mathfrak{R}, v > 0 \quad (3.30)$$

and  $h(x)$  is differentiable, convex and strictly positive. More generally, c-programming can handle problems where

$$f(v, w) := \frac{g(v)}{h(w)}, \quad v, w \in \mathfrak{R} \quad (3.31)$$

where  $g(v)$  is differentiable, concave and non-negative.  $h(w)$  is differentiable, convex and strictly positive.

### 3.2.2 Multiplicative Functions

In consideration of multiplicative functions, we are considering the composite functions of the following form

$$f(v, w) := g(v)h(w) \quad (3.32)$$

where both  $g(v)$  and  $h(w)$  are real-valued differential functions.  $f(v, w)$  is pseudoconcave if either one of the following conditions holds [5]:

1.  $g(v)$  is nonnegative and concave and  $h(w)$  is positive and concave.
2.  $g(v)$  is nonpositive and convex and  $h(w)$  is negative and convex.

The multiplicative function and the ratio function are essentially of the same structure, ie (3.31) can be expressed as a product form

$$f(v, w) := g(v) \frac{1}{h(w)} \quad (3.33)$$

and (3.32) can be rewritten in a fractional form

$$f(v, w) := \frac{g(v)}{1/h(w)}. \quad (3.34)$$

### 3.2.3 Additive Functions

It has to be pointed out that just because a function made up of the sum of two pseudoconcave functions does not necessarily make it pseudoconcave, the composite function

$$f(v, w) := g(v) + h(w), \quad v, w \in \mathfrak{R} \quad (3.35)$$

where both  $g(v)$  and  $h(w)$  are pseudoconcave, is not necessarily pseudoconcave. But since concavity entails pseudoconcavity and furthermore concavity is preserved under addition. C-programming will accept the additive form from (3.35) if  $g(v)$  and  $h(w)$  are both differentiable and concave.

# Chapter 4

## Composite Concave Linear Programming

In the previous chapter we looked at the formulation of Composite Concave Problems. In this chapter we look at Composite Concave Linear Programs (CCLP), which is a subclass of Composite Concave Programming Problems. Composite Concave Linear Programs is used in situations where the constraints of the optimization problem are linear and the objective function is a composite of linear functions.

### 4.1 Formulation

Composite Concave Linear Programming is the area of global optimization that deals with the formulation, modeling, analysis of problems with the following basic form:

*Problem CCLP*

$$z^* := \min_{x \in X} f(x) := g(u(x)) \quad (4.1)$$

where

$X \in \mathfrak{R}^n$  is defined by a system of linear constraints

$u$  is a linear function on  $X$  with values in  $\mathfrak{R}^k$

$g$  is a real valued function on  $u(X) := \{u(x) : x \in X\}$ . The basic assumption is that  $g$  is differentiable and concave with respect to  $u(x)$ .

Optimal solutions to this nonlinear optimization problem are obtained via the following parametric problem:

*Problem CCLP*( $\lambda$ )

$$z^*(\lambda) := \min_{x \in X} \lambda^T u(x), \quad \lambda \in \Lambda \subset \mathfrak{R}^k \quad (4.2)$$

where  $\Lambda$  is an appropriate subset of  $\mathfrak{R}^k$ . The key observation is that under the above conditions, if *Problem CCLP* has an optimal solution then there is a  $\lambda^o \in \mathfrak{R}^k$  such that every optimal solution to *Problem CCLP*( $\lambda^o$ ) is also optimal for *Problem CCLP*.

**Example 4.1** Consider the nonlinear programming problem

$$z^* := \min_x \sqrt{x_1 + 2x_2 + x_3} + \sqrt{3x_1 + 5x_2 + 2x_3} \quad (4.3)$$

$$3x_1 + 4x_2 + 3x_3 = 12 \quad (4.4)$$

$$6x_1 + 2x_2 + 3x_3 \geq 11 \quad (4.5)$$

$$x_1, x_2, x_3 \geq 0 \quad (4.6)$$

we can set  $n = 3$  and  $k = 2$ . letting

$$u(x) = (x_1 + 2x_2 + x_3, 3x_1 + 5x_2 + 2x_3) \quad (4.7)$$

$$g(u(x)) = \sqrt{u_1(x)} + \sqrt{u_2(x)} \quad (4.8)$$

This leads to the associated parametric problem taking the form:

$$z^*(\lambda) := \min_x \lambda_1(x_1 + 2x_2) + \lambda_2(3x_1 + 5x_2), \quad \lambda \in \Lambda \quad (4.9)$$

$$3x_1 + 4x_2 + 5x_3 = 12 \quad (4.10)$$

$$4x_1 + 2x_2 + 3x_3 \geq 10 \quad (4.11)$$

$$x_1, x_2, x_3 \geq 0 \quad (4.12)$$

where  $\Lambda$  is an appropriate subset of  $\mathfrak{R}^2$ .

## 4.2 Optimality Conditions

We now identify some sufficient conditions for the existence of an optimal  $\lambda$ , for the parametric *Problem CCLP*( $\lambda$ ), and outline a procedure for solving our target

*Problem CCLP.* And with the identification of these conditions, it is then necessary to identify a subset  $\Lambda$  containing at least one optimal  $\lambda$ . And when this is done, we then only have to solve our parametric problem for  $\lambda \in \Lambda$  and retrieve the solution to our target problem.

In this section we concentrate on the sufficient conditions for the existence of an optimal  $\lambda$ , this will be done notation wise in the framework of *Concave Programming* as is more convenient. Let us consider the *Concave Programming* format:

$$\text{Problem } P : \quad z^* := \min_{x \in X} f(x), \quad X \subseteq \mathfrak{R}^n \quad (4.13)$$

$$\text{Problem } P(\lambda) : \quad z^* := \min_{x \in X} \lambda^T x, \quad \lambda \in \mathfrak{R}^n \quad (4.14)$$

where

$X^*$  denotes the set of optimal solutions to *Problem P*

$X^*(\lambda)$  denotes the set of optimal solutions to *Problem P*( $\lambda$ )

$\lambda \in \mathfrak{R}^n$ .

It has to be noted that

The only condition on  $X$  is that  $X \subseteq \mathfrak{R}^n$ .

The only condition on  $f$  is that it is a real valued function on  $X$ .

**Theorem 4.2** [6] *Assume that  $f$  is differential and concave on some open convex set containing set  $X$ . Then,*

$$X^*(\nabla f(x^*)) \subseteq X^*, \forall x^* \in X^* \quad (4.15)$$

This theorem says that if  $f$  is differentiable and concave on some convex set containing set  $X$ , then an optimal solution to the *Problem P*( $\nabla f(x^*)$ ) is also a optimal solution to *Problem P*. If  $x^*$  is an optimal to *Problem P* then  $\lambda^* = \nabla f(x^*)$  is an optimal  $\lambda$ .

Theorem 4.2 provides us with simple but also general sufficient conditions for the existence of a optimal  $\lambda$ .

1. The objective function  $f$  is *differentiable* on some open convex set containing  $X$ .
2. The objective function is *concave* on this set.
3. The target problem, *Problem P*, possesses at least one *optimal solution*.

In translating this theorem from that for *Problem P* to that of *Problem CCLP*, the following *corollary* is obtained

**Corollary 4.3** [6] *Assume in the framework of Problem CCLP the outer function  $g$  is differentiable and concave on some open convex set containing  $u(X) := x \in X$ . Then,*

$$X^*(\nabla g(u(x^*))) \subseteq X^*, \forall x^* \in X^*. \quad (4.16)$$

In words, as long as the conditions of differentiability and convexity are met, if  $x^*$  is an optimal solution of the target problem, *Problem CCLP*, then  $\lambda = \nabla g(u(x^*))$  is an optimal  $\lambda$ .

#### EXAMPLE

We shall now look at the proof of theorem 4.2, but first we must state two important properties of concave functions.

<p>Some Properties of Concave Functions</p> <p><math>f</math> is a real valued function on <math>X \subseteq \mathfrak{R}^n</math></p> <p><math>v, w \in X</math></p> <p><math>f(\alpha v + (1 - \alpha)w) \geq \alpha f(v) + (1 - \alpha)f(w), \forall 0 \leq \alpha \leq 1</math></p> <p><math>f(w) \leq f(v) + (w - v)^T \nabla f(v)</math></p>
--

Table 4.1: Properties of concave functions

The first equation in table 4.1 requires  $X$  to be convex, the second requires  $f$  to be differentiable on some open convex subset of  $\mathfrak{R}^n$  containing  $X$ . The second condition implies that

$$f(w) \geq f(v) \Rightarrow w^T \nabla f(v) w \geq v^T \nabla f(v) \quad (4.17)$$

$$f(w) \leq f(v) \Leftarrow w^T \nabla f(v) \leq v^T \nabla f(v) \quad (4.18)$$

#### Proof of Theorem 4.2

Let  $\hat{x}$  be any optimal solution to the *Problem P*. Then, by definition  $f(x) \geq f(\hat{x}), \forall x \in X$ . Therefore (4.17) entails that  $x^T \nabla f(\hat{x}) \leq \hat{x}^T \nabla f(\hat{x})$ . It follows then that

$$\hat{x} \in X^* \Rightarrow \hat{x} \in X^*(\nabla f(\hat{x})) \quad (4.19)$$

Now, let  $x^o$  be any optimal solution to *Problem P*( $\nabla f(x^*)$ ). Then from (4.18) it follows that

$$y^o \in X^*(\nabla f(x^*)) \Rightarrow x^o \in X^*. \quad (4.20)$$

This implies that  $X^*(\nabla f(x^*)) \subseteq X^*. \square$

**Theorem 4.4** [6] *The Fundamental theorem of CCLP*

Assume that in the framework of Problem CCLP the outer function  $g$  is differentiable and pseudoconcave on some open convex set containing  $u(X) := \{u(x) : x \in X\}$ . Then,

$$X^*(\nabla(u(x^*))) \subseteq X^*, \forall x^* \in X^*. \quad (4.21)$$

In summary, the following is a simple test for the existence of an optimal  $\lambda$  for a CCLP problem:

1. There is an open convex subset of  $\Re^k$  containing  $u(X)$  such that  $g$  is differentiable on this set?
2. Is  $g$  pseudoconcave on this set?
3. Does Problem CCLP possess an optimal solution?

If you can answer YES to these three questions, then an optimal  $\lambda$  exists. Actually, if this is the case, then all the elements of the set

$$\Lambda^* := \{\nabla g(u(x^*)) : x^* \in X^*\} \quad (4.22)$$

are optimal  $\lambda$ 's. But it has to be pointed out that this set does not automatically contain all the optimal  $\lambda$ 's.

### 4.3 Search Space

In the previous section we spelled out optimality conditions for the existence of an optimal to the Problem CCLP. As already has been pointed out, this problem is solved by the parametric Problem CCLP( $\lambda$ ) for an optimal value of  $\lambda$ .

But showing that an optimal  $\lambda$  exists for a given problem is one thing, going ahead and finding this optimal is a whole other problem. This is because there could be infinitely many values for  $\lambda$  and searching all of these may not be the greatest of ideas.

Our plan to combat this is to construct a set  $\Lambda \subseteq \Re^k$  containing at least one optimal  $\lambda$  and find the solution of Problem CCLP( $\lambda$ ) for all  $\lambda \in \Lambda$ .

The set  $\Lambda$  should be small so the solving of the parametric problem can be done quickly, but at the same time we do not want to spend a lot of time constructing this set. These two considerations often lead to a trade off between the size of the  $\Lambda$  and the effort needed in construction it.

Following is a list of scenarios in the framework of which such a space might have to be constructed.

- You may have been given the solution for the optimal  $\lambda$ , take it and run as this is unlikely to occur. The *Problem CCLP* can be solved with one instance of the *Problem CCLP*( $\lambda$ ) with one run through of your LP package.
- A  $\Lambda$  is given. Maybe it has come to your attention that the optimal  $\lambda$  is in a given  $\Lambda \subseteq \Re^n$ .
- Construct an outer-approximation to  $\{\nabla g(u(x)) : x \in X\}$ . There is often a quick way to construct a set containing  $\{\nabla g(u(x)) : x \in X\}$  that may be larger but still effective as a search space.
- Set  $\Lambda = \Re^k$ . Is to be used when the other options are no good.

Consider the problem form before, with the equality replaced by inequality.

**Example 4.5**

$$z^* := \min_x \sqrt{x_1 + 2x_2 + x_3} + \sqrt{3x_1 + 5x_2 + 2x_3} \quad (4.23)$$

$$3x_1 + 4x_2 + 3x_3 \leq 12 \quad (4.24)$$

$$6x_1 + 2x_2 + 3x_3 \geq 11 \quad (4.25)$$

$$x_1, x_2, x_3 \geq 0 \quad (4.26)$$

Now formally set the problem as follows to comply with the format for *Problem CCLP*( $\lambda$ )

$$f(x) = \sqrt{x_1 + 2x_2 + x_3} + \sqrt{3x_1 + 5x_2 + 2x_3} \quad (4.27)$$

$$u(x) = (x_1 + 2x_2 + x_3, 3x_1 + 5x_2 + 2x_3) \quad (4.28)$$

$$g(u(x)) = \sqrt{u_1(x)} + \sqrt{u_2(x)} \quad (4.29)$$

and let  $X$  denote the subset of  $\Re^3$  defined by the linear system specified by (4.24)-(4.26). Differentiating  $g$  we obtain

$$\nabla g(u(x)) = \left( \frac{\partial}{\partial y_1} g(y), \frac{\partial}{\partial y_2} g(y) \right) \Big|_{y=u(x)} \quad (4.30)$$

$$= \left( \frac{1}{2\sqrt{y_1}}, \frac{1}{2\sqrt{y_2}} \right) \Big|_{y=(x_1+2x_2+x_3, 3x_1+5x_2+2x_3)} \quad (4.31)$$

$$= \left( \frac{1}{2\sqrt{x_1 + 2x_2 + x_3}}, \frac{1}{2\sqrt{3x_1 + 5x_2 + 2x_3}} \right) \quad (4.32)$$

Hence,

$$\Lambda^\circ := \left\{ \left( \frac{1}{2\sqrt{x_1 + 2x_2 + x_3}}, \frac{1}{2\sqrt{3x_1 + 5x_2 + 2x_3}} \right) : x \in X \right\} \quad (4.33)$$

It is possible to obtain upper and lower bounds for  $u(x)$  by minimizing and maximizing  $u_1(x)$  and  $u_2(x)$  over  $X$ . This requires us to solve 4 LP problems, and you don't really want to do this, do you?

We could alternatively construct a rough bound - the third suggestion mentioned above, with the use of the constraints in this case. Consider (4.24). If we make all of the co-efficients in (4.24) greater or equal to the corresponding co-efficients in  $u_1(x)$ , we are able to obtain an upper bound for  $u_1(x)$ . So if we multiple (4.24) by  $1/2$  and  $3/2$  (for  $u_2(x)$ ) respectively,

$$\frac{3}{2}x_1 + 2x_1 + \frac{3}{2} \leq 6 \quad (4.34)$$

$$\frac{9}{2}x_1 + 6x_1 + \frac{9}{2} \leq 18 \quad (4.35)$$

$$(4.36)$$

Since all the decision variables are non-negative, we can safely say that the following holds

$$u_1(x) = x_1 + 2x_2 + x_3 \leq 6 \quad (4.37)$$

$$u_2(x) = 3x_1 + 5x_2 + 2x_3 \leq 18 \quad (4.38)$$

We can also do the same sort of thing and find an upper bound for  $u(x)$ . Divide (4.25) by 6 and 2

$$x_1 + 2x_2 + 3x_3 \geq \frac{11}{6} \quad (4.39)$$

$$3x_1 + x_2 + 3x_3 \geq \frac{11}{2} \quad (4.40)$$

Since there is no negative, this implies that

$$u_1(x) = x_1 + 2x_2 + x_3 \geq \frac{11}{6} \quad (4.41)$$

$$u_2(x) = 3x_1 + 5x_2 + 2x_3 \geq \frac{11}{2} \quad (4.42)$$

Therefore, the set

$$\Lambda = \left[ \frac{1}{2\sqrt{6}}, \frac{1}{2\sqrt{11/6}} \right] \times \left[ \frac{1}{2\sqrt{18}}, \frac{1}{2\sqrt{11/2}} \right] \quad (4.43)$$

is an outer approximation of  $\Lambda^\circ$ .

## 4.4 Algorithms

### 4.4.1 Mid-Point Algorithm

This algorithm is based on the *Mid-Point Enterprise*. We are going to consider the 2-dimensional case of this algorithm in this section. The algorithm is designed for the CCLP problems where the objective function is of the form  $f(x) = g(u_1(x), u_2(x))$ . The two dimensional parametric problem is converted into an equivalent single parameter problem

$$\text{Problem } CCLP(\lambda) := z^*(\lambda) := \min_{x \in X} \lambda u_1(x) + u_2(x), \underline{\lambda} \leq \lambda \leq \bar{\lambda} \quad (4.44)$$

where  $X \subseteq \Re^n$  is a system of linear constraints.  $\Lambda = [\underline{\lambda}, \bar{\lambda}]$ . At the completion of the algorithm, the interval  $\Lambda$  is split into subintervals, each interval corresponding to a feasible solution of the problem. These subintervals are made with the use of the *Mid-Point Rule*:

given a quartet  $((\alpha, x), (\beta, y))$  we can compute the next estimate for  $\lambda$

$$\hat{\lambda} = \mu(((\alpha, x), (\beta, y)))) := \frac{y_2 - x_2}{x_1 - y_1} \quad (4.45)$$

this new point is in the interval  $[\alpha, \beta]$  and splits into two subintervals  $[\alpha, \hat{\lambda}]$  and  $[\hat{\lambda}, \beta]$ . At each iteration of the algorithm there are two possibilities:

The interval  $[\alpha, \beta]$  is discarded because it is already covered by some  $x \in X$ .

The interval is split in two subintervals, namely  $[\alpha, \hat{\lambda}]$  and  $[\hat{\lambda}, \beta]$ .

The algorithm ends when all the intervals/subintervals have been discarded.

This algorithm can be summarized as follows:

#### Mid Point CCLP Algorithm

**Step 0:**

Solve *Problem CCLP*( $\lambda$ ) for  $\lambda = \underline{\lambda}$  and  $\bar{\lambda}$  and set  $Q = (((\underline{\lambda}, x), (\bar{\lambda})))$  where  $x = SOLVE(\underline{\lambda})$  and  $y = SOLVE(\bar{\lambda})$ , and set  $z^o = \min\{f(x), f(y)\}$ . Let  $x^o$  denote the best of  $x$  and  $y$ .

**While**  $Q \neq \emptyset$ , **DO:**

**Step 1:**

Let  $q = ((\alpha, x), (\beta, y))$  be the first quartet in  $Q$ , set  $\hat{\lambda} = \mu(q)$  and  $\hat{x} = SOLVE(\hat{\lambda})$  and remove  $q$  from  $Q$ .

**Step 2:**

If  $q$  failed the *Mid-Point Test* Do:

If  $\hat{x} \notin X^*(\beta)$  add  $((\beta, y), (\hat{\lambda}, \hat{x}))$  to  $Q$

If  $\hat{x} \notin X^*(\alpha)$  add  $((\alpha, x), (\hat{\lambda}, \hat{x}))$  to  $Q$

**Step 3:**

If  $f(\hat{x}) < z^o$  set  $z^o = f(\hat{x})$  and  $x^o = \hat{x}$ .

Note:  $SOLVE(\hat{\lambda})$  means solve  $\hat{\lambda}$  using your LP package and retrieve solution to  $x$ .

In the next chapter we implement the *Mid-Point Algorithm*, with the use of 'Xpress-IVE - student edition' to solve the linear model formed by *Problem CCLP*( $\lambda$ ).



# Chapter 5

## Numerical Example

In this chapter we are solving a Composite Concave Linear Program using the *Mid-Point Algorithm* discussed in the previous chapter with the assistance of a normal Linear Programming package (Xpress-IVE).

Consider the problem

$$z^* := \min_{x \in X} \sqrt{4x_1 + x_2 + 3x_3} + \log(10x_1 + x_2 + 3x_4 + 4x_5 + 5) \quad (5.1)$$

where  $X \subset \mathbb{R}^5$  is defined by

$$3x_1 + 4x_2 + 2x_3 + 5x_4 + 2x_5 \geq 16 \quad (5.2)$$

$$2x_1 + 3x_2 + 4x_3 + 5x_4 + 3x_5 \geq 20 \quad (5.3)$$

$$4x_1 + 2x_2 + 3x_3 + 5x_4 + 5x_5 \geq 18 \quad (5.4)$$

$$5x_1 + 4x_2 + 3x_3 + 5x_4 + 5x_5 \geq 22 \quad (5.5)$$

$$2x_1 + 5x_2 + 4x_3 + 2x_4 + 4x_5 \geq 18 \quad (5.6)$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0 \quad (5.7)$$

We tackle this problem using the Mid-Point algorithm for 2-dimensional problems. This algorithm is based on the *Mid-Point Enterprise*. This algorithm is designed for CCLP problems where the objective function is of the form  $f(x) = g(x_1, x_1)$ , as is shown below

$$f(x) = g(u(x)) = \sqrt{4x_1 + x_2 + 3x_3} + \log(10x_1 + x_2 + 3x_4 + 4x_5 + 5) \quad (5.8)$$

$$u(x) = (4x_1 + x_2 + 3x_3, 10x_1 + x_2 + 3x_4 + 4x_5 + 5) \quad (5.9)$$

This leads to the following parametric problem:

$$\begin{aligned} & \textit{Problem CCLP}(\lambda) : \\ & z^*(\lambda_1, \lambda_2) := \min_{x \in X} \{ \lambda_1 u_1(x) + \lambda_2 u_2(x) \} \end{aligned} \quad (5.10)$$

Then dividing by  $\lambda_2$ , now setting  $\lambda = \frac{\lambda_1}{|\lambda_2|}$ . the two dimensional parametric problem is converted into the equivalent single-parameter problem

$$z^*(\lambda) := \min_{x \in X} \{ \lambda u_1(x) + u_2(x) \}, \quad \lambda \in [0, \infty) \quad (5.11)$$

subject to the following constraints

$$3x_1 + 4x_2 + 2x_3 + 5x_4 + 2x_5 \geq 16 \quad (5.12)$$

$$2x_1 + 3x_2 + 4x_3 + 5x_4 + 3x_5 \geq 20 \quad (5.13)$$

$$4x_1 + 2x_2 + 3x_3 + 5x_4 + 5x_5 \geq 18 \quad (5.14)$$

$$5x_1 + 4x_2 + 3x_3 + 5x_4 + 5x_5 \geq 22 \quad (5.15)$$

$$2x_1 + 5x_2 + 4x_3 + 2x_4 + 4x_5 \geq 18 \quad (5.16)$$

$$u_1(x) = 4x_1 + x_2 + 3x_3 \quad (5.17)$$

$$u_2(x) = 10x_1 + x_2 + 3x_4 + 4x_5 + 5 \quad (5.18)$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0 \quad (5.19)$$

Firstly, check that this problem satisfies the sufficient conditions for a *Composite Concave Linear Programming Problem* according to theorem 4.4 (Fundamental theorem of CCLP).

The function  $g$  is differentiable and pseudoconcave since  $\sqrt{u_1(x)}$  is concave and also  $\log(u_2(x))$  is concave, and since  $g(u(x))$  is additive. Using the property that two concave functions when added together form a concave function, hence are also pseudoconcave.  $g(u(x))$  is also differentiable, since

$$\nabla g(u(x)) = \left( \frac{\partial}{\partial y_1} g(y), \frac{\partial}{\partial y_2} g(y) \right) \Big|_{y=u(x)} \quad (5.20)$$

$$= \left( \frac{1}{2\sqrt{y_1}}, \frac{1}{y_2} \right) \Big|_{y=(4x_1+x_2+3x_3, 10x_1+x_2+3x_4+4x_5+5)} \quad (5.21)$$

$$= \left( \frac{1}{2\sqrt{4x_1+x_2+3x_3}}, \frac{1}{10x_1+x_2+3x_4+4x_5+5} \right) \quad (5.22)$$

So  $g(u(x))$  is differentiable on the set  $0 < u(x) \subset \mathfrak{R}^2$

Now it is time to begin solving *Problem CCLP*( $\lambda$ ).

With consideration of the set  $\Lambda$ , since  $u(x) > 0$ , we are safely saying that the optimal  $\lambda$  take values in  $[0, \infty)$ .

Solve the parametric problem for  $\lambda = 0$  and  $\lambda = \infty$

$$q = ((0, (24, 5)), (\infty, (0, 13)))$$

$$Q = ((0, (24, 5)), (\infty, (0, 13)))$$

$$z^o = \min\{f(24, 5), f(0, 13)\} = f(0, 13) = 2.5649$$

$$x^o = (0, 0, 0, 0, 8)$$

Now select the first quartet in  $Q$ , namely  $q = ((0, (24, 5)), (\infty, (0, 13)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\hat{\lambda} = \frac{y_2 - x_2}{x_1 - y_1} \tag{5.23}$$

$$= \frac{13 - 5}{24 - 0} = \frac{1}{3} \tag{5.24}$$

Remove  $q$  from  $Q$ , now  $Q = \{ \}$ , and solve for  $\lambda = 1/3$ , resulting in  $\hat{u} = (9.43, 8.43)$ . The  $q$  fails the *Mid-Point Test*, since  $\hat{u} \notin X^*(0)$  and  $\hat{u} \notin X^*(\infty)$ . And update

$$Q = \{(0, (24, 5)), (1/3, (9.43, 8.43)), (1/3, (9.43, 8.43)), (\infty, (0, 13))\}$$

$$f(\hat{u}) = f(9.43, 8.43) = 6.5084 > z^o = 2.5649, \text{ no updating required}$$

$$x^o = (0, 0, 0, 0, 8)$$

$Q$  is not empty, so continue.

Select the first quartet in  $Q$ ,  $q = ((0, (24, 5)), (1/3, (9.43, 8.43)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned} \hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{8.42857 - 5}{24 - 9.42857} = 0.2353 \end{aligned} \tag{5.25}$$

Remove  $q$  from  $Q$ , now  $Q = \{(1/3, (9.43, 8.43)), (\infty, (24, 5))\}$ , and solve for  $\lambda = 0.2353$ , resulting in  $\hat{u} = (19, 6)$ . The  $q$  fails the *Mid-Point Test*, since  $\hat{u} \notin X^*(0)$  and  $\hat{u} \notin X^*(1/3)$ . And update

$$Q = \{(0, (24, 5)), (0.2353, (19, 6)), (0.2353, (19, 6)), (1/3, (9.43, 8.43)), (1/3, (9.43, 8.43)), (\infty, (0, 13))\}$$

$$f(\hat{u}) = f(19, 6) = 6.5084 > z^o = 2.5649, \text{ no updating required}$$

$$x^o = (0, 0, 0, 0, 8)$$

$Q$  is not empty, continue

Select the first quartet in  $Q$ ,  $q = ((0, (24, 5)), (0.2353, (19, 6)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{6 - 5}{24 - 19} = \frac{1}{5}\end{aligned}\tag{5.26}$$

Remove  $q$  from  $Q$ , now  $Q = \{(0.2353, (19, 6)), (1/3, (9.43, 8.43)), (1/3, (9.43, 8.43)), (\infty, (0, 13))\}$ , and solve for  $\lambda = 1/5$ , resulting in  $\hat{u} = (24, 5)$ . The  $q$  passes the *Mid-Point Test*, since  $\hat{u} \in X^*(0)$ . This implies that there is no further search in the quartile  $q$  required.

$Q$  is not empty, continue.

Select the first quartet in  $Q$ ,  $q = ((0.2353, (19, 6)), (1/3, (9.43, 8.43)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{8.42857 - 6}{19 - 9.42857} = 0.2537\end{aligned}\tag{5.27}$$

Remove  $q$  from  $Q$ , now  $Q = \{(1/3, (9.43, 8.43)), (\infty, (24, 5))\}$ , and solve for  $\lambda = 0.2537$ , resulting in  $\hat{u} = (11, 8)$ . The  $q$  fails the *Mid-Point Test*, since  $\hat{u} \notin X^*(0.2353)$  and  $\hat{u} \notin X^*(1/3)$ . Update

$$Q = \{(0.2353, (19, 6)), (0.2537, (11, 8)), (0.2537, (11, 8)), (1/3, (9.43, 8.43)), (1/3, (9.43, 8.43)), (\infty, (24, 5))\}$$

$$f(\hat{u}) = f(11, 8) = 5.3961 > z^o = 2.5649, \text{ no updating required}$$

$$x^o = (0, 0, 0, 0, 8)$$

$Q$  is not empty, continue.

Select the first quartet in  $Q$ ,  $q = ((0.2353, (19, 6)), (0.2537, (11, 8)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{8 - 6}{19 - 11} = 1/4\end{aligned}\tag{5.28}$$

Remove  $q$  from  $Q$ , now  $Q = \{(0.2537, (11, 8)), (1/3, (9.43, 8.43)), (1/3, (9.43, 8.43)), (\infty, (0, 13))\}$ , and solve for  $\lambda = 1/4$ , resulting in  $\hat{u} = (11, 8)$ . The  $q$  passes the *Mid-Point Test*, since  $\hat{u} \in X^*(0.2537)$ . This implies that there is no further search in the quartile  $q$  required.

$Q$  is not empty, continue.

Select the first quartet in  $Q$ ,  $q = ((0.2537, (11, 8)), (1/3, (9.43, 8.43)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{8.43 - 8}{11 - 9.43} = 0.2727\end{aligned}\tag{5.29}$$

Remove  $q$  from  $Q$ , now  $Q = \{(1/3, (9.43, 8.43)), (\infty, (0, 13))\}$ , and solve for  $\lambda = 0.2727$ , resulting in  $\hat{u} = (9.43, 8.43)$ . The  $q$  passes the *Mid-Point Test*, since  $\hat{u} \in X^*(1/3)$ . This implies that there is no further search in the quartile  $q$  required.

$Q$  is not empty, continue.

Select the first quartet in  $Q$ ,  $q = ((1/3, (9.43, 8.43)), (\infty, (0, 13)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{13 - 8.43}{9.43 - 0} = 0.4848\end{aligned}\tag{5.30}$$

Remove  $q$  from  $Q$ , now  $Q = \{\}$ , and solve for  $\lambda = 0.4848$ , resulting in  $\hat{u} = (4/3, 35/3)$ . The  $q$  fails the *Mid-Point Test*, since  $\hat{u} \notin X^*(1/3)$  and  $\hat{u} \notin X^*(\infty)$ . Update

$$\begin{aligned}Q &= \{(1/3, (9.43, 8.43)), (0.4848, (4/3, 35/3)), (0.4848, (4/3, 35/3)), (\infty, (0, 13))\} \\ f(\hat{u}) &= f(4/3, 35/3) = 3.114 > z^o = 2.5649, \text{ no updating required} \\ x^o &= (0, 0, 0, 0, 8)\end{aligned}$$

$Q$  is not empty, continue.

Select the first quartet in  $Q$ ,  $q = ((1/3, (9.43, 8.43)), (0.4848, (4/3, 35/3)))$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{35/3 - 8.43}{9.43 - 4/3} = \frac{2}{5}\end{aligned}\tag{5.31}$$

Remove  $q$  from  $Q$ , now  $Q = \{(0.4848, (4/3, 35/3)), (\infty, (0, 13))\}$ , and solve for  $\lambda = 2/5$ , resulting in  $\hat{u} = (4/3, 35/3)$ . The  $q$  passes the *Mid-Point Test*, since  $\hat{u} \in X^*(0.4848)$ . This implies that there is no further search in the quartile  $q$  required.

$Q$  is not empty, continue.x

Select the first quartet in  $Q$ ,  $q = ((0.4848, (4/3, 35/3)), (\infty, (0, 13)),)$  and apply the *Mid-Point Rule* to select new  $\lambda$

$$\begin{aligned}\hat{\lambda} &= \frac{y_2 - x_2}{x_1 - y_1} \\ &= \frac{13 - 35/3}{4/3 - 0} = 1\end{aligned}\tag{5.32}$$

Remove  $q$  from  $Q$ , now  $Q = \{\}$ , and solve for  $\lambda = 1$ , resulting in  $\hat{u} = (4/3, 35/3)$ . The  $q$  passes the *Mid-Point Test*, since  $\hat{u} \in X^*(0.4848)$ . This implies that there is no further search in the quartile  $q$  required.

$Q$  is empty, STOP.

The following table gives a summary of results

$\lambda$	$u(x)$	$x$	$f(x)$	$z^*(\lambda^1)$
[0,0.2353]	(24,5)	(0,0,8,0,0)	6.508	5
(0.2353,0.2537]	(11,8)	(0,2,3,0,1)	5.396	10.4707
(0.2537,1/3]	(9.4286,8.4286)	(0,2.1429,2.4286,0,1.2857)	6.508	10.7907
(1/3,0.4848]	(4/3,35/3)	(0,4/3,0,0,16/3)	3.611	11.5714
(0.4848, $\infty$ )	(0,13)	(0,0,0,0,8)	2.565	12.3131

<sup>1</sup> the value of  $\lambda$  used for this calculation is the lower bound

Table 5.1:

Figure 5.1, shows the break points of the algorithm. The five distinct solutions to the problem and the values of  $\lambda$  they correspond to. Table 5.2 gives the value of

$\lambda$	0	1/5	0.235	1/4	0.254	0.273	1/3	2/5	0.485	1	$\infty$
$z^*(\lambda)$	5.00	9.80	10.47	10.75	10.79	11.00	11.57	12.20	12.31	13.00	13.00

Table 5.2:

$z^*(\lambda)$  for all  $\lambda$  solved in the *Mid-Point Algorithm*.

In summary, the algorithm required 11 different values of  $\lambda$  to solve the problem to an optimal, in doing so, there is shown to be 5 distinct solutions.

The optimal was found to be at  $x = (0, 0, 0, 0, 8)$  with an objective value of  $z^* = 2.5648$ .

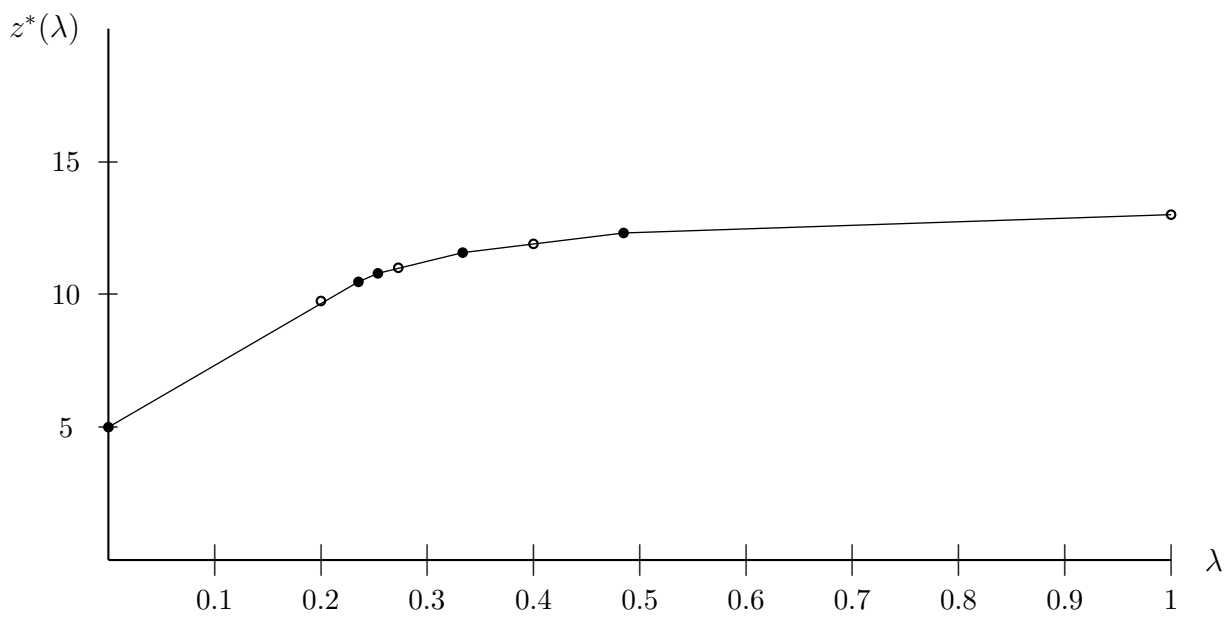


Figure 5.1:

• indicate the break points of the  $\lambda$ 's



# Chapter 6

## Conclusion/Discussion

In this thesis we have investigated Composite Concave Programming with the major focus on Composite Concave Linear Programming. We were able to solve a problem of the CCLP with the use of a Linear Programming package (Xpress-IVE, student edition), and find the global optimal for this problem.

We investigated the formulation of CCLP's and more generally CCP's, with respect to composite functions and the linearization of the problems to make them acceptable to LP packages for solving. The finding of a search space for the  $\lambda$  in the search for a optimal and the trade off between finding a tight interval for  $\lambda$  and larger interval.

We acquired sufficient conditions for the existence of optimal solution to these particular problems.

With regard to further study, it would be possible to investigate the solving of higher order problems with the use of LP software as I only looked at the case when  $k = 2$ . We could investigate more real world problems and applications.

With regard to comments, find in Xpress-IVE student edition if there is a setting that gives you a warning that the problem is unbounded, as it didn't point this out and gave some nonsense answers.



# Chapter 7

## Appendix

### 7.1 Appendix A

```
Xpress-IVE code model CCLP2
uses "mmxprs"; !gain access to the Xpress-Optimizer solver
```

```
!optional parameters section
!parameters
! SAMPLEPARAM1="c: test "
! SAMPLEPARAM2=false
!end-parameters
```

```
!sample declarations section
declarations
! ...
i = 1..5
j = 1..2
!numbers: array(i) of real
x:array(i) of mpvar
u:array(j) of mpvar
LL = 10000 !lambda
```

```
end-declarations
!OBJECTIVE FUNCTION
```

```

total:=LL*u(1)+u(2)

! declare constraints
3*x(1)+4*x(2)+2*x(3)+5*x(4)+2*x(5) >= 16
2*x(1)+3*x(2)+4*x(3)+5*x(4)+3*x(5) >= 20
4*x(1)+2*x(2)+3*x(3)+5*x(4)+5*x(5) >= 18
5*x(1)+4*x(2)+3*x(3)+5*x(4)+5*x(5) >= 22
2*x(1)+5*x(2)+4*x(3)+2*x(4)+4*x(5) >= 18
x(1) >= 0
x(2) >= 0
x(3) >= 0
x(4) >= 0
x(5) >= 0
u(1) = 4*x(1)+x(2)+3*x(3)
u(2) = 5+10*x(1)+x(2)+3*x(4)+x(5)
! end declarations

minimize(total)

writeln("lambda = ",LL)
writeln("z*(lambda) = ",getobjval)
writeln("x(1) = ",getsol(x(1)))
writeln("x(2) = ",getsol(x(2)))
writeln("x(3) = ",getsol(x(3)))
writeln("x(4) = ",getsol(x(4)))
writeln("x(5) = ",getsol(x(5)))

writeln("u(1) = ",getsol(u(1)))
writeln("u(2) = ",getsol(u(2)))

end-model

```

## 7.2 Appendix B

### Xpress-IVE output

lambda = 0  
z\*(lambda) = 5  
x(1) = 0  
x(2) = 0  
x(3) = 8  
x(4) = 0  
x(5) = 0  
u(1) = 24  
u(2) = 5

lambda = 10000  
z\*(lambda) = 13  
x(1) = 0  
x(2) = 0  
x(3) = 0  
x(4) = 0  
x(5) = 8  
u(1) = 0  
u(2) = 13

lambda = 0.333333  
z\*(lambda) = 11.5714  
x(1) = 0  
x(2) = 2.14286  
x(3) = 2.42857  
x(4) = 0  
x(5) = 1.28571  
u(1) = 9.42857  
u(2) = 8.42857

lambda = 0.2353  
z\*(lambda) = 10.4707  
x(1) = 0  
x(2) = 1  
x(3) = 6  
x(4) = 0  
x(5) = 0  
u(1) = 19  
u(2) = 6

lambda = 0.2  
z\*(lambda) = 9.8  
x(1) = 0  
x(2) = 0  
x(3) = 8  
x(4) = 0  
x(5) = 0  
u(1) = 24  
u(2) = 5

lambda = 0.2537  
z\*(lambda) = 10.7907  
x(1) = 0  
x(2) = 2  
x(3) = 3  
x(4) = 0  
x(5) = 1  
u(1) = 11  
u(2) = 8

lambda = 0.25  
z\*(lambda) = 10.75  
x(1) = 0  
x(2) = 2  
x(3) = 3  
x(4) = 0  
x(5) = 1  
u(1) = 11  
u(2) = 8

lambda = 0.2727  
z\*(lambda) = 10.9997  
x(1) = 0  
x(2) = 2  
x(3) = 3  
x(4) = 0  
x(5) = 1

u(1) = 11  
u(2) = 8

lambda = 0.4848  
z\*(lambda) = 12.3131  
x(1) = 0  
x(2) = 1.33333  
x(3) = 0  
x(4) = 0  
x(5) = 5.33333  
u(1) = 1.33333  
u(2) = 11.6667

lambda = 0.4  
z\*(lambda) = 12.2  
x(1) = 0  
x(2) = 1.33333  
x(3) = 0  
x(4) = 0  
x(5) = 5.33333  
u(1) = 1.33333  
u(2) = 11.6667

lambda = 1  
z\*(lambda) = 13  
x(1) = 0  
x(2) = 1.33333  
x(3) = 0  
x(4) = 0  
x(5) = 5.33333  
u(1) = 1.33333  
u(2) = 11.6667

## 7.3 Appendix C

### Mid-Point Test

Let  $((\alpha, x), (\beta, y))$  be any quartet, let  $\gamma = \mu((\alpha, x), (\beta, y))$  and consider the following two cases:

Case 1: either  $x \in X^*(\gamma)$  and/or  $y \in X^*(\gamma)$

Case 2: neither  $x \in X^*(\gamma)$  nor  $y \in X^*(\gamma)$

Then,

In Case 1:  $x \in X^*(\lambda), \forall \lambda \in co(\alpha, \beta)$  and  $y \in X^*(\lambda), \forall \lambda \in co(\beta, \gamma)$ .

In Case 2:  $x$  and  $y$  do not cover  $co(\alpha, \beta)$ , that is  $co(\alpha, \beta) \not\subseteq \Lambda^*(x) \cup \Lambda^*(y)$ .

# Bibliography

- [1] Sniedovich, M. “*Dynamic Programming, (appendix c)*”, (Marcel Dekker, New York. 1992)
- [2] Churilov, L; Bomze, I, M; Sniedovich, M; Ralph, D., “Hyper Sensitivity Analysis of Portfolio Optimization Problems,” *Asia-Pacific Journal of Operations Research*, **Vol. 21** (No.3), (World Scientific Publishing Co. & Operations Research Society of Singapore. 2004): 297-317.
- [3] Byrne, A; Sniedovich, M; Churilov, L, “Handling soft constraints via composite concave programming,” *Journal of the Operational Research Society*, (Operational Research Society Ltd, 1998).
- [4] Sniedovich, M; Macalalag, E; Findlay, S, “The Simplex Method as a Global Optimizer: A C-Programming Perspective,” *Journal of Global Optimization*, **Vol 4**, (Netherlands: Kluwer Academic Publishers, 1994): 89-109.
- [5] Avriel, M, “Nonlinear Programming: Analysis and Methods,” **Englewood Cliffs, New Jersey**: 1976
- [6] Sniedovich, M: ”A Very Gentle Introduction to Composite Concave Linear Programming: How to get more out of your beloved LP Solver via CCLP,” : 2006
- [7] Macalalag, E and Sniedovich, M, ”On the Importance of being a Sensitive LP Package”, *Preprint Series No. 3-1991*,(Department of Mathematics, The University of Melbourne, 1991)
- [8] Markowitz HM, ”The general mean variance portfolio selection problem. In Mathematical models in Finance”, SD Howison, FP Kelly and P Wilmott (eds), (London, UK: Chapman and Hall): 1995
- [9] Sniedovich, M, ”Dynamic Programming”, (New York: Marcel Dekker): 1992
- [10] Winston, WL, ”Operations Research: Applications and Algorithms”, (Duxbury Press, Belmont, California,USA): 1994
- [11] Costa, A, ”620-361 Operations Research Techniques and Algorithms, Course Notes”, (The University of Melbourne):2005

- [12] Macalalag, E; Sniedovich, M, "Generalized Linear Programming and Sensitivity Analysis Techniques", *Naval Research Logistics*, **Vol. 43**, (John Wiley & Sons, Inc 1996): 397-413