SPATIAL DYNAMIC PROGRAMMING ON THE WATER TREATMENT PROBLEM

by

JONG-WOEI WHANG

B.S. (Industrial Engineering)

National Tsing Hua Univerisity

Hsin Chu, Taiwan, Republic of China, 1978

A MASTER'S THESIS

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

Kansas State University

Manhattan, Kansas

1983

Approved by :

[illegible]

Major Professor

ACKNOLEDGEMENTS

The author wishes to express his sincere appreciation to his major advisor, Dr. E. S. Lee, for his valuable guidance and creative advice during the preparation of this thesis.

He also wishes to thank Dr. Chi-lung Huang and Dr. Doris Grosh for serving on his committee.

The author then wishes to extend his deep gratitude to Emily's parents, Mr. Chi-Ti Chen, and his wife Stella, for their constant encouragement and inspiration.

Finally, he would like to dedicate this work to his dear mother for her understanding and support.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

CHAPTER 1

AN INTRODUCTION TO LARGE-SCALE SYSTEMS

1.1 Introduction

Just in the last decade has major research and development of large scale systems taken form. At least five distinct industries, aerospace, computing, power, semiconducter, and communications-- call for the response of recent activity in the large-scale systems area [Palmer, 1982]. Besides these, there are other such systems which will eventually present a great deal of challenge, such as urban traffic network, digital communication network, ecological systems, and economic systems [Sandell, 1978]. Although the theory of large-scale systems has been studied by Pearson, Tamura, and others over the past ten years, the field of large-scale systems and their decentralized control is still in its infancy [Singh, 1980]. Because the computational difficulties grow exponentially as the scale of a system increases, some special techniques are needed to deal with large-scale systems [Araki, 1978]. The usual procedures for optimizing large-scale systems are decomposition and coordination. To decompose a system means to divide a whole system into smaller ones, i.e., to make the system behavior more tractable. Yet, it is necessary to add some coordinate constraints to recover the original structure before decomposition. In adding these constraints, the model selected, or the algorithms implemented, will determine whether the result is global optimum or not. These same determiners will also show if the optimization process is stable or unstable.

1.2 Decomposition and decentralized control of a large-scale system

A large-scale system can be viewed as many interconnected subsystems, within which the interaction can be static or dynamic. The overall system consists of many input variables, output variables, and state variables. Input variables can be controlled to regulate both the state variables and output variables. Also, performance of the criteria for the large-scale system can be optimized by means of regulation over the control variables. An interconnected dynamical system is shown in Fig. 1.2.1. In this figure, blocks of the same row represent variables in the same subsystem but at different time instants. Only subsystems K and K+1 at time t , $t+t_0$, $t+2t_0$ are shown in this figure.

Due to the complexity and large dimension of the total system, it is impossible to optimize it unless the decomposition method is applied. This task becomes feasible through the decentralized control algorithm [Li, 1982]. Decentralization implies that the various controllers in the system are allowed only to measure some outputs and to regulate some inputs of the system, thus providing decentralized control over the entire system [Ozguner, 1982

].

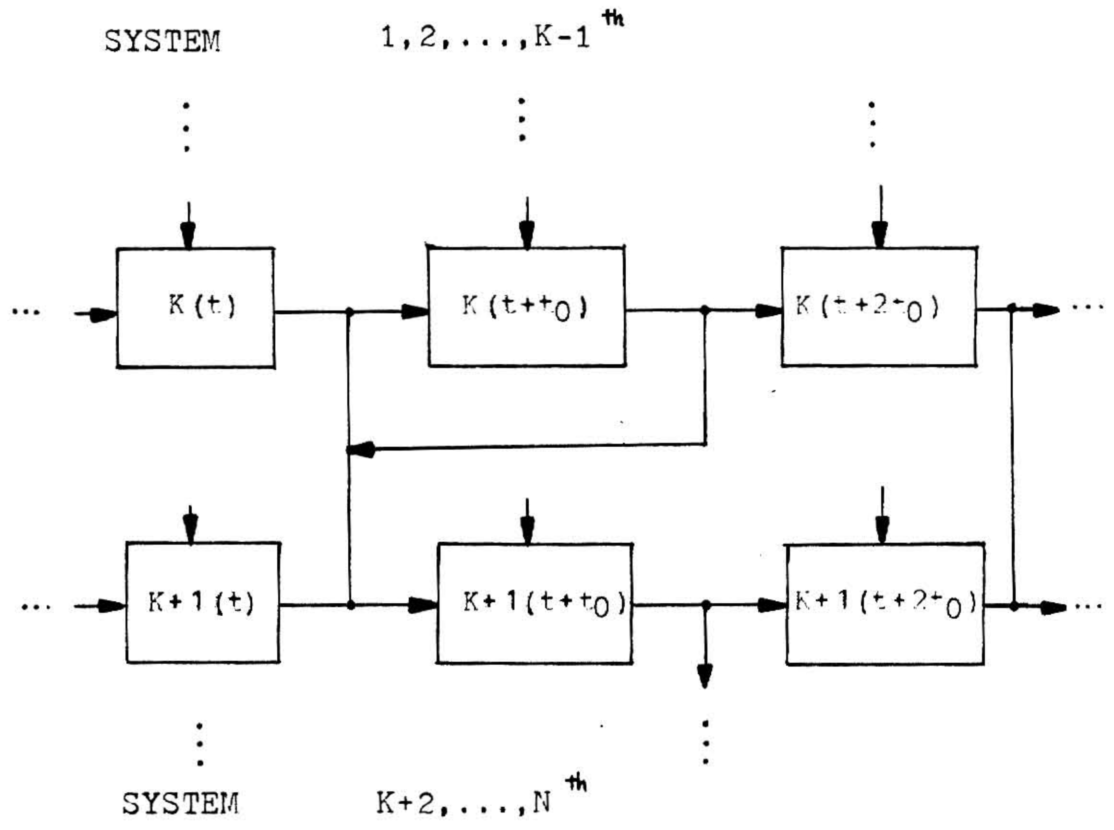


Fig. 1.2.1 : An Interconnected Dynamical System

1.3 Approaches and algorithms for optimization of large-scale system

The decomposition technique was originated by Dantzig-Wolfe for linear programming problems [Dantzig, 1979]. Later followed Arrow and Hurwicz's paper [Arrow, 1960], Takahara's algorithm [Takahara, 1964] on optimal control, and Lasdon etc.'s paper [Lasdon, 1965] on 'separable' mathematical programming. There has been an increasing interest in the so called 'two-level' optimization algorithms. The basic idea in algorithms of this sort is to separate whole problems into many 'independent' subproblems and then solve the subproblems individually. Meanwhile, it takes into account the interaction between subsystems by use of coordinate constraints, thus dividing the large system into smaller ones. The price paid here is that it might be necessary to solve the same problems several times via some iterative schemes. This kind of technique serves as a starting point for the theory of large-scale system [Cohen, 1978], [Dantzig, 1979].

Since then, there have been several frameworks developed to solve large-scale system problems through different approaches. The first class of methods includes the aggregation method [Howard, 1971] and the perturbation method [Kwathy, 1977] of model simplification. The first of these methods assumes that a simplified model can be obtained by introducing a coarser state space description, while the latter method ignores some dynamic interactions within the system in order to simplify the model. The second class of methods is the decentralized feedback control and the multilevel method for deterministic control [Sandel, 1978].

Most of the decomposition schemes for large-scale system fall into two categories: either 'hierarchical system formalism' or 'approximation procedures'. The first catagory includes goal coordination, model coordination, and interaction principle, whereas the latter includes successive approximate dynamic programming, singular perturbation techniques ,and the N -coupling technique, [Mesarovic, 1970], [Wismer, 1971], [Mahmoud, 1977], [Larson, 1970], [Kokotivic, 1971]. Of these latter two techniques, both require an iterative scheme to obtain the solution. And in most cases certain conditions have to be met in order to guarantee convergence to the optimal solution [Cline, 1977].

Of all these techniques stated in the last paragraph, dynamic programming is the most powerful tool one can use to solve problems with multi-stage decision processes, which is a common phenomenon of large-scale system. Also, the basic approach of the dynamic programming technique may be suitable for implementing on a digital computer. However, the usefulness of dynamic programming is severely limited by the dimensionality of the problem. Despite the more advanced capability of modern computing facilities, many practical problems still can not be solved by dynamic programming due to the existing computational capabilities limit. This is called the 'dimensionality difficulty' [Bellman, 1962]. Frequently, some of the decomposition techniques have been used to overcome this difficulty. This is explained in the following section.

1.4 Spatial dynamic programming on large-scale system

One new approach of the dynamic programming technique intended to solve problems with high dimensionality was developed by Robert E. Larson in 1977 [Cline, 1977]. This new approach, called 'spatial dynamic programming' (SDP), can treat non-serial structured problems , as well as highly interacted systems, more easily and efficiently. The basic idea of the spatial dynamic programming technique is to treat the overall system as many smaller subsystems upon which dynamic programming can apply. One requirement of SDP is that the optimization problem itself has to be at least weakly decomposable. 'Weak decomposibility' means that, given the original problem as

$$J_N(\underline{0}) = \min_{(\underline{u}_1,\ldots,\underline{u}_N)} F(\underline{u}_1,\underline{u}_2,\ldots,\underline{u}_N)$$

assume the values of some decision vectors u are given, for instance if

$$(\underline{u}_{K+1},\ldots,\underline{u}_N) = (\hat{\underline{u}}_{K+1},\ldots,\hat{\underline{u}}_N)$$

$$\text{then} \quad J_K(\hat{\underline{u}}_{K+1},\ldots,\hat{\underline{u}}_N) = \min_{(\underline{u}_1,\ldots\underline{u}_N)} F(\underline{u}_1,\underline{u}_2,\ldots,\underline{u}_N) \qquad (1.3.1)$$

is still well defined for all K, K=1,2,3...,N

This property implies that the original problem can be divided into many subproblems without disturbing the original problem structure. This characteristic provides the basis for SDP. The details of the SDP technique will be explained in the following chapter.

CHAPTER 2

INTRODUCTION TO SPATIAL DYNAMIC PROGRAMMING

2.1 Preliminary conditiona to apply spatial dynamic programming

Although spatial dynamic programming is a promising tool to treat large-scale systems, some assumptions are needed about the problem of concern. These assumptions are much more general than other optimization techniques which may require the problem to be linear, quadratic and so on. The preliminary conditions needed to apply spatial dynamic programming are demonstrated in the following section.

2.2 Separable, weakly decomposable, strongly decomposable functions

Suppose a function $F(x \; , x \; , \ldots , x \;)$ can be defined by subfunctions f_i , i.e. ,

$$F_1(x_1) = x^0 \qquad x^0 : \text{ initial condition}$$

$$F_2(x_2, x_1) = f_2(x_2, F_1(x_1))$$

$$\vdots$$

$$F_{N-1}(x_{N-1}, \ldots, x_1) = f_{N-1}(x_{N-1}, F_{N-2}(x_{N-2}, \ldots, x_1))$$

$$F_N(x_N, \ldots, x_1) = f_N(x_N, F_{N-1}(x_{N-1}, \ldots, x_1))$$

$$F(x_N, \ldots, x_1) = F_N(x_N, \ldots, x_1)$$

where x° represents the initial conditions

And if F_i is well defined for every i, then F is separable [Cline, 1977].

The f_i are the separating functions of F . Note that F need not be linear to be separable. This implies prohlems of linear, quardratic, and even nonlinear forms could be treated if they are separable. Furthermore, given that F is separable, and all its separating functions f_i (x,y) are non-decreasing with respect to y, or increasing with respect to y, then F is said to be weakly decomposable and strictly decomposable respectively. If all f_i (x,y) are nondecreasing with respect to both x and y, then F is strongly decomposable. The definitions given above are summerized in the following table :

Table 2.2.1 Definition of decomposability

F is separable, f(x,y) are separating functions of F		
All f_i (x+y)	non-decreasing	increasing
depends on y only	weakly decomposable	strictly decomposable
depends on both x and y	strongly decomposable	******

The above definitions are very important in finding the necessary conditions for the problems in which the SDP method can be applied. The details of the decomposable condition will be explained in the following paragraph.

Assume the most general form of the mathematical programming problem can be defined as follows :

$$\text{OPT} \quad F(X_1, X_2, \ldots, X_N) \qquad (X_1, X_2, \ldots, X_N) \in S$$

where OPT : the maximization or minimization according to a spesific cost or performance criterion

S : represents the set of variables which satisfy constraints

(2.2.1)

Although the X's can be vectors, here they are assumed scalers for simplicity.

Instead of using the form above, the problem can also be presented in a 'separable form', to implement the decomposition procedure. That is :

$$\text{OPT} \; F_N(f_N(x_N), f_{N-1}(x_{N-1}), \ldots, f_1(x_1)) \tag{2.2.2}$$

subject to :

$$\begin{array}{lll} G_m(g_{Nm}(x_N), g_{N-1m}(x_{N-1}), \ldots, g_{1m}(x_1)) \quad 0 & & m=1,2,\ldots,L \\ G_m(g_{Nm}(x_N), g_{N-1m}(x_{N-1}), \ldots g_{1m}(x_1)) = 0 & & m=L+1,\ldots,M \\ g_{nm}(x_n) <= 0 & & m=M+1,\ldots,K \\ & & n=1,2,\ldots,N \end{array}$$

In this more 'concrete' form, constraints have been divided into three catagories, because they will be treated differently. An 'individual' constraint (not coupled with other variables), such as : $g_{nm}(x_n) <= 0$, $m=M+1,M+2,\ldots,K$, can be transformed without loss of generality into these sets : $X = \left\{ x : g_{nm}(x_n) <= 0,\ m=M+1,M+2,\ldots,K \right\}$. This will simplify the optimizing procedure by considering the feasible region of the variable x without involving other constraints and/or variables. The above problem is referred to as the principal problem and can be abbreviated as the following :

$$P = \mathrm{OPT} \left\{ \begin{array}{l} F(f_N(x_N), f_{N-1}(x_{N-1}), \ldots, f_1(x_1)) : \\ G_m(g_{Nm}(x_N), g_{N-1m}(x_{N-1}), \ldots) <= 0 \\ \quad m=1,2,\ldots,L \\ G_m(g_{Nm}(x_N), g_{N-1m}(x_{N-1}), \ldots) = 0 \\ \quad m=L+1,L+2,\ldots,M \\ \text{and} \quad x \in X \quad , \ n=1,2,\ldots N \end{array} \right\} \tag{2.2.3}$$

It is time to find the sufficient conditions for decomposition in both principal objective functions and principal constrant functions. Under these conditions, principal problems can be decomposed into subproblems by dynamic programming. Therefore the principal problem can be solved by combining iterative solutions to subproblems by dynamic programming. Consider the real valued functions, H_n $(n=1,2,\ldots,N)$ and h_n $(n=2,3,\ldots)$. H_n is defined in the subset of E^n, while h_i's are defined in the subsets of E^2. This is shown in the following diagram. (Fig. 2.2.1)

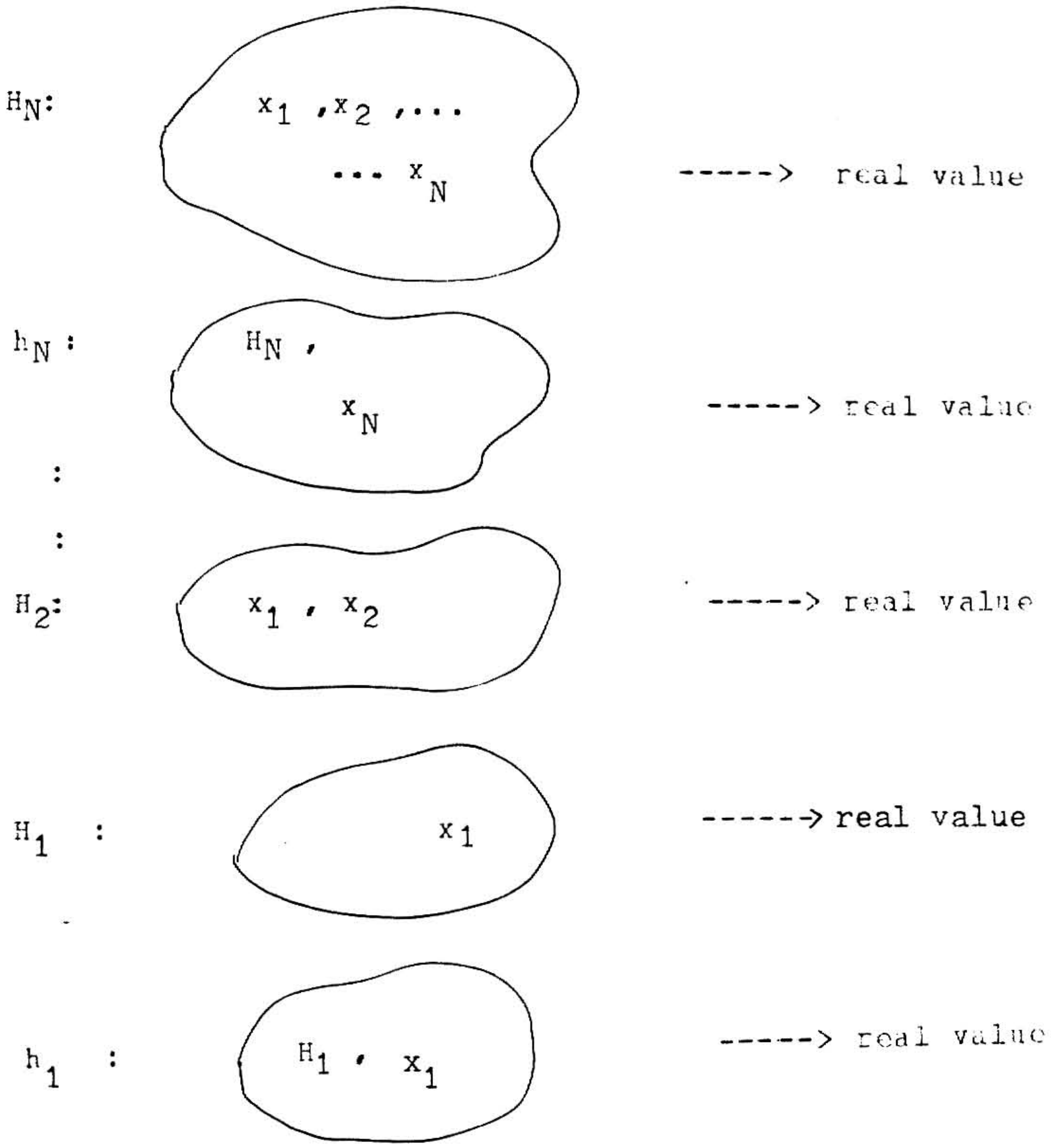


Fig. 2.2.1 Schetmetic representation of function H and its separating function h

The formal definition can be stated as follows :

Def. The real value function H is said to be separate if and only if there exist real value functions h_n ($n=2,3,\ldots,N$) such that

$$H(y_N,y_{N-1},\ldots,y_1) = h_N(y_N,h_{N-1}(y_{N-1},\ldots,h_2(y_2,H_1(y_1)..)$$

(2.2.4)

That is, if it is separable, H is represented as follows :

$$H(y_N,y_{N-1},\ldots,y_1) = H_N(y_N,y_{N-1},\ldots,y_1)$$

(2.2.5)

while H_n ($n=2,\ldots,N$) are iteratively defined as :

$$H_n(y_n,y_{n-1},\ldots,y_1) = h_n(y_n,H_{n-1},(y_{n-1},\ldots,y_1))$$

(2.2.6)

The above functions h_i's are referred to as the separating functions of H. For example, suppose that

$$H(x,y,z) = (x^2 + \log(20 + y)^{1/2})\ e^{-z} \qquad (2.2.7)$$

Here H is a separable function if it can be defined as :

$$H(x,y,z) = H_3(x,y,z)$$

$$= h_3(g_3(z), H_2(y,x))$$
$$= h_3(g_3(z), h_2(g_2(y), H_1(x)))$$

$$g_3(t_3) = e^{-t_3}$$
$$h_3(t_1, t_2) = t_1 + t_2$$

$$H_2(t_1, t_2) = (t_1^2 + \log(20 + t_2))^{1/2}$$
$$h_2(t_1, t_2) = (t_1 + t_2)^{1/2}$$
$$g_2(t_2) = \log(20 + t_2)$$
$$H_1(t_1) = t_1^2$$

It may be possible to express H as different groups of iteratively related separating functions; that is, H can be represented as

$$H = H_N(x_N, x_{N-1}, \ldots, x_1) \quad \text{or}$$
$$H = H_M(y_M, y_{M-1}, \ldots, y_1) \quad \text{with different } h_n\text{'s and } h_m\text{'s}$$

In this situation, grouping the variables and choosing different functions to 'separate' the principal objective and principal constraint functions is an arbitrary job, unless the effectiveness of the decomposition and/or composition algorithm is taken into consideration. This fact can be demonstrated with an example as follows :

If $H(w,x,y,z) = w + x + y^2 + z$,

then $H(w,x,y,z)$ can be represented as :

$$H(w,x,y,z) = (w + x + y^2) + z$$
$$\text{or} \quad = (w + x + z) + y^2$$
$$\text{or} \quad = (w + x) + (y^2 + z) \qquad (2.2.8)$$

This separating procedure depends on the dimensionality for each group of variables, which is essentially based on the idea of the reducing dimensionality of subproblems as much as possible. The third method stated in (2.2.8) has proved its effectiveness on the parallel processing algorithm.

Besides the separable property shown above, the dynamic programming decomposition technique requires other conditions on the principal objective function, as well as constraint functions which would guarantee a 'smooth' operation on the decomposition procedure. These are : weakly decomposable, strongly decomposable, strictly decomposable and left-continuous conditions.

The real valued function H is said to be weakly decomposable if H is separable and all separating functions $h_n(y,z)$ are nondecreasing with respect to z, which is the 'cost-to-go' or 'performance-to-go' arguemant. If H is separable and all its separating functions $h_n(y,z)$ are non-decreasing with respect to both y and z , then H is said to be strongly decomposable. Note here that weak decomposibility is more general than strong decomposibility because if a function is strongly decomposable with respect to both y and z, then it has to be decomposable with respect to z. For example

$$h(y,z) = 0 \qquad \text{for } y \text{ and } z < 0$$

$$= y + z \qquad \text{for } y \text{ or } z >= 0$$

In short, $h(y,z) = y + z$ is strongly and hence weakly decomposable, but $h(y,z) = \sin y + z$ is only weakly decomposable.

The separable function H is defined as strictly decomposable if separating functions h_n of H are increasing rather than decreasing. This property is preferred in dynamic programming because it guarantees an equal or better suboptimal solution from one stage to another. For example

$$\begin{aligned} & H_n(y_n, y_{n-1}, \ldots, y_1) \\ &= h_n(y_n, H_{n-1}(y_{n-1}, \ldots, y_1)) \quad <= Z_{nm} \\ & H_{n-1}(y_{n-1}, y_{n-2}, \ldots, y_1) \\ &= h_{n-1}(y_{n-1}, H_{n-2}(y_{n-2}, \ldots, y_1)) \quad <= Z_{n-1\,m} \end{aligned} \qquad (2.2.10)$$

The difference between Z_{nm} and Z_{n-1m} is guaranteed by the strictly decomposable property of h_{n-1} and h_n, which assures the suboptimal solution can be improved stage by stage.

A function $f(x)$ is said to be left-continuous at point x if $f(x)$ approaches $f(x^-)$ as x approaches x from the left. That is

$$\lim_{x \to x^-} f(x) = f(x^-) \qquad (2.2.11)$$

The separable function H is said to be left-continuous when all of its separating functions are left-continuous. That is

$$\lim_{\substack{y \to y^- \\ z \to z^-}} h(y,z) = h(y^-, z^-)$$

(2.2.12)

This property is essential in the computational aspect of dynamic programming bcause of its implication that the function is smooth. This is demonstrated by an example

Assuming the constraint $h_n(y,z) <= Z$, is the upper bound of $h_n(y, z) = z$, and assuming the existing (y_0, z_0) satisfies $h_n(y_0, z_0) = z$, then in searching the possible value of y and z from small to large value, the following property (2.2.13) is desired :

$$\lim_{\substack{y \to y^- \\ z \to z^-}} h(y,z) = h(y^-, z^-) = Z$$

(2.2.13)

Without these assumptions, it would be impossible to know when to cease the searching process.

In the above, the weak or strong decomposibiltiy has been defined as the nondecreasing or increasing property. Left-continuity is defined as the continuity approach from the left. These definitions, however, can be extended to the monotony, or strict monotony, as well as the right-continuity of the separating functions, based on the same idea. All the theorem and corollary are valid as is, or with only a minor modification. This can be verified in the process of devising conditions for decomposibility.

2.3 conditions for decomposability

To begin with, suppose the principal constraint functions G_m $(m=1,2,\dots,L)$ are weakly decomposable and left-continuous with separating functions h_{nm} $(n=2,3,..,N\ ,m=1,2,\dots,L)$ It is necessary to consider inequality constraints first :

$$G_{Nm}(g_{Nm}(x_N),g_{N-1m}(x_{N-1}),\dots,g_{1m}(x_1)) <= Z_m \qquad (2.3.1)$$

Since G_{Nm} isweakly decomposable and left-continuous, then,

$$\begin{aligned} &G_{Nm}(g_{Nm}(x_N),g_{N-1m}(x_{N-1}),\dots,g_{1m}(x_1)) \\ &= h_{Nm}(g_{Nm}(x_N),G_{N-1m}(g_{N-1m}(x_{N-1}),\dots,g_{1m}(x_1)) <= Z_m \end{aligned}$$

$$(2.3.2)$$

Usually this constraint will not be satisfied by all values of $g_{nm}(X_N)$, but only by some real numbers. Let V_{Nm} $(m=1,2,\dots,L)$ be the set of these real numbers, and denote its elements by y_{Nm} , then $y_{Nm} \in V_{Nm}$, while $V_{Nm} \subset R(g_{Nm})$, where R is the range function values. Proceeding with this extension, the function becomes :

$$\begin{aligned} &h_{Nm}(g_{Nm}(x_{Nm}\ ,h_{N-1m}(g_{N-1m}(x_{N-1}\ ,\dots,h_{1m}(g_{1m}(x_1))\dots) \\ &= G_{Nm}\ (g_{N-1m}(x_{N-1}),\dots,g_{1m}(x_1))\dots) \\ &= Z_m \end{aligned} \qquad (2.3.3)$$

This function may also represent

$$h_{nm}(g_{nm}(x_n), G_{n-1m}(g_{n-1m}(x_{n-1}), \ldots, g_{1m}(x_1))..) = Z_{nm}$$

If $q_{nm}(x_n) = y_{nm}$ is used, then there exists a real number Z', to satisfy :

$$h_{nm}(y_{nm}, Z') = Z_{nm}$$

(2.3.4)

In other words, for $h_{nm}(y_{nm}, Z') <= Z_{nm}$, where $y_{nm} \in V_m$, it is impossible to find Z' to satisfy this constraint, also, for different $y_{nm} \in V_{nm}$, a different Z' may be found to satisfy the same constraint. For example, let

$$h(y,z) = y^2 + z^2 = a$$

$$\text{if } a = 0 \text{ then } V = \{ \ 0 \ \}$$

$$\text{if } a >= 0 \text{ then } V = \{ \ y \ ; -a^{1/2} <= y <= a^{1/2} \ \}$$

(2.3.5)

Therefore z can always be found for the specific value of a and y, which satisfy the constraint, by selecting z from

$$\underline{Z} = \{ z \ ; \ - (a - y^2)^{1/2} <= z <= (a - y^2)^{1/2} \}$$

In selecting different upper bounds of constraint for different stages, the following two consecutive stages are used for demonstration.

$$h_{nm}\ (y_{nm}\ ,Z') = Z_{nm} \qquad y_{nm} \in V_{nm}$$

$$h_{n-1m}(y_{n-1m},Z\ ) = Z_{n-1m} \qquad y_{n-1m} \in V_{nm}$$

(2.3.6)

If one finds Z satisfying $h_{nm}\ (\ y_{nm}\ ,Z'\ )\ =\ Z_{nm}$ then all $\underline{Z} = \{ z\ ;\quad z <= Z' \}$ will satisfy the constraints because h_{nm} is non-decreasing in Z' . It is possible to find a maximum in Z to satisfy the constrant. is shown below :

$$Z_{n-1m} = \max \ \{ Z'\ ;\ \ h_{nm}(y_{nm},Z') = Z_{nm}\}$$

then $h_{nm}(g_{nm}(x_n),G_{n-1m}(g_{n-1m}(x_{n-1}),g_{n-1m}(x_{n-1}),g_{1m}(x_1))..) <= z_{nm}$

$y_{nm} = g_{nm}(x_n)$ and G_{n-1m} is bounded by $\max \ \{ Z' \}$

The reason for finding the maximum Z' is

$$G_{n-1m}(g_{n-1m}(x_{n-1}),g_{n-2m}(x_{n-2}),\ldots,g_{1m}(x_1))\ldots)$$

$$= h_{n-1m}(g_{n-1m}(x_{n-1}),G_{n-1m}(g_{n-1m}),\ldots,g_{1m}(x_1))\ldots)$$

$$<= Z_{n-1m} = \max \ \{ Z' \}$$

(2.3.7)

Here the technique is illustrated by a numerical example ,

Objective : $\max H\ (u_3,u_2,u_1) = u_3^3\ u_2 u_1^2$

subject to : $u_1^2 + u_2^2 + u_3^2 \qquad = \ 12$

(2.3.8)

Furthermore, if :

$$H(u_3,u_2,u_1) = H_3(f_3(u_3),f_2(u_2),f_1(u_1))$$

$$G(u_3,u_2,u_1) = G_3(g_3(u_3),g_2(u_2),g_1(u_1))$$

then H and G may be represented as :

$$\begin{aligned} & H_3(f_3(u_3),f_2(u_2),f_1(u_1)) \\ = \; & h_3(f_3(u_3),H_2(f_2(u_2),f_1(u_1))) \\ = \; & h_3(f_3(u_3),h_2(f_2(u_2),H_1(f_1(u_1)))) \end{aligned} \tag{2.3.9}$$

They may also be represented as :

$$\begin{aligned} & G_3(g_3(u_3),g_2(u_2),g_1(u_1)) \\ = \; & h_3'(g_3(u_3),G_2(g_2(u_2),g_1(u_1))) \\ = \; & h_3'(g_3(u_3),h_2'(g_2(u_2),G_1(g_1(u_1)))) \end{aligned} \tag{2.3.10}$$

These examples illustrate that H and G are separable, when converting the constraint

$$u_1^2 + u_2^2 + u_3^2 \leqq 12 \tag{2.3.11}$$

into a standard 'separated' form, it becomes

$$h_3(u_3^2 + h_2'(u_1^2, u_2^2)) = Z_3 \tag{2.3.12}$$

Here Z_3 is equal to 12. By assuming $y_3 = \hat{u}_3^2 \in V_3 \subset R(u_3^2)$, it is easy to find Z_2 when $Z_3(=12)$ and y_3 are fixed. In this case, let $Z_2 = \max \{ Z'; \; h'(y_3, Z') <= Z_3 \}$, where $\hat{u}_3^2 + Z' = Z_3$, $Z' = Z_3 - \hat{u}_3^2$ and $\max \{ Z' \} \leqq Z_3 - \hat{u}_3^2 = Z_2$ are given.

Although the value of $\hat{u}_3^2$ is unknown for the time being, it is still possible to search for Z_3. That is, for all the possible value of Z, this feature provides the basis of computational aspect of dynamic programming. To demonstrate the decomposition recursiveness of principal constraint functions more clearly, two sets will be introduced : S and s. Small s implies the parameter of the current step, while the large S represents the steps already finished.

$$S_n(Z_n) = \left\{ (x_n, x_{n-1}, \ldots, x_1); h_n(g_n(x_n), G_{n-1}(g_{n-1}(x_{n-1}), \ldots, g_1(x_1))) = z_n,\ x_i \in X_i\ (i=1,2,\ldots,n) \right\}$$

(2.3.13)

While X_i are feasible region for x_i, $S_n(Z_n)$ has one more element than its proceeding counterpart $S_{n-1}(Z_{n-1})$. This gap is filled recursively by small s_n, which is defined as

$$s_n = \{ x_n : g_n(x_n) = y_n,\ x \in X_n \}$$

(2.3.14)

However, S does not satisfy the constraint for all y_n. Only $y \in V$ will make the product meaningful; therefore, the followng relation is obtained : (U : union)

$$S_n(Z_n) = \bigcup_{y_n \in V_n} s_n(y_n) \times S_{n-1}(Z_{n-1})$$

(2.3.15)

If h_n is also non-decreasing with respect to $g_n(x_n)$, the relationship is then defined as : $S_n(y_n) = \{x_n : g_n(x_n) <= y_n,$ and $x_n \in Xn \}$ From a computational viewpoint, this is effective because the of upper bound of the decision space can be obtained. This definition will also apply to the principal equality constraint function. For example , let

$$G_m(g_{Nm}(x_N), g_{N-1m}(x_{N-1}), \ldots, g_{1m}(x_1)) = Z \quad (m= M+1, \ldots, L)$$

then define :

$$S_n(Z_n) = \{ (x_n, x_{n-1}, \ldots, x_1) ; h_n(x_n, g_n(x_n), G_{n-1m}(g_{n-1m}(x_{n-1}), \ldots, g_{1m}(x_1))) = Z_n, \ x_i \in X_i \} \tag{2.3.16}$$

According to the discussions stated above, all the constraints can be summarized in the following forms :

$$\begin{array}{lll} G_m(m=1,2,\ldots,K) <= Z_m & \text{is strongly decomposable} \\ & \text{and left continuous,} \\ G_m(m=K+1,\ldots,L) = Z_m & \text{is weakly decomposable} \\ & \text{and left continuous} \\ G_m(m=L+1,\ldots,M) = Z_m & \text{is strictly decomposable} \end{array} \tag{2.3.17}$$

This is to say that :

$$u_3(u_2 + u_1) <= z_{11}, \ h_{31}(x,y) = xy, \ h_{21}(x,y) = x + y$$
$$\cos(u_3 + \sin(u_1 + u_2)) <= z_{12}, \ h_{32}(x,y) = \cos(x+y)$$
$$h_{22}(x,y) = \sin(x + y), \ h_{33}(x,y) = x + y , \ h_{23}(x,y) = x+y$$

$$u_3^2 + u_2^2 + u_1^2 = Z_{13} \tag{2.3.18}$$

When searching for state variables (Z's) and decision variables (u's) which are constrained by the first class of constraints, one will find that

$u_2 + u_1 <= Z_{21}$, then select $Z_{21} = \max \{ Z' ; Z' = Z_{11}/u_3 \}$ for the specific value of u_3

(2.3.19)

Here Z_{21} is the upper bound of state space for the next step. Meanwhile, searching the space of decision variables is also bounded by some values, such as :

$$u_3 = \hat{u}_3 = \max \{ u' ; u' = Z_{11} /(u_1 + u_2) \} \tag{2.3.20}$$

The upper bound for both state and decision variables exists because of strong decomposibility. For equality constraints, decision variables have been searched in almost the same manner as inequality constraints. The only difference is that the state variable of the next step is decided by selecting decision variables at a current step as equal, but not equal or less than some specific Z value. For example if $u^2 <= 1$ then u may be searched from 1, 0.9, 0.8,..., but for $u^2 = 1$, the u value can only be selected as 1 or -1, which makes a difference from the computational aspect. Therefore the general form of S would be :

$$S_n(Z_n) = \{ (x_n, x_{n-1}, \dots, x_1); \\ G_{nm}(g_{nm}(x_n), g_{n-1m}(x_{n-1}), \dots, g_{1m}(x_1) = Z_{nm}, \\ (m=1,1,\dots,L), \\ G_{nm}(g_{nm}(x_n), g_{n-1m}(x_{n-1}), \dots, g_{1m}(x_1) = Z_{nm} \\ (m=L+1,\dots,M), \quad x_i \in X_i \} \qquad (2.3.21)$$

Inequality constraints may still be grouped into these two classes :

$G_{nm}(g_{nm}(x_n), g_{n-1m}(x_{n-1}), \dots, g_{1m}(x_1)) <= Z_{nm}$
$(m=1,2,\dots,K)$, which are strongly decomposable, and
$G_{nm}(g_{nm}(x_n), g_{n-1m}(x_{n-1}), \dots, g_{1m}(x_1)) <= Z_{nm}$
$(m=K+1,\dots,L)$, which are weakly decomposable.

Thus the form of S_n will be written as :

$$S_n(\underline{Z}_n) = \bigcap_{m=1}^{M} S_{nm}(Z_{nm}) \qquad M : \text{total number of constraints}$$

$$= \bigcup_{\underline{y}_n \in \underline{V}_n} \left\{ \bigcap_{m=1}^{M} S_{nm}(y_{nm}) \times \bigcap_{m=1}^{M} S_{n-1m}(Z_{n-1m}) \right\}$$

$$= \bigcup_{\underline{y}_n \in \underline{V}_n} \left\{ \left[\left(\bigcap_{m=1}^{K} S'nm(y_{nm}) \right) \cap \left(\bigcap_{m=K+1}^{L} S_{nm}(y_{nm}) \right) \right] \times \left(\bigcap_{m=1}^{L} S_{n-1m}(Z_{n-1m}) \right) \right\} \cup \left[\left(\bigcap_{m=L+1}^{K} S_{nm}(y_{nm}) \right) \times \left(\bigcap_{m=L+1}^{M} S_{n-1m}(Z_{n-1m}) \right) \right] \Big\}$$

where $V_n = \overset{M}{\underset{m=1}{X}} V_{nm}$ (2.3.22)

Here the principal problem becomes

$$P \equiv \text{OPT}\left\{F\,(f_N(x_N), f_{N-1}(x_{N-1}), \ldots, f_1(x_1)); \; (x_N, x_{n-1}, \ldots, x_1) \in S_N(\underline{Z}_N)\right\} \tag{2.3.23}$$

Thus far, only the decomposition of principal constraint functions have been discussed; however, now decomposition of the objective functions will be considered.

Suppose the principal objective function is strongly decomposable with separating functions h_n $(n=2,3,\ldots,N)$. One must define $F_n(n=1,2,\ldots,N)$ in a separable manner as with principal constraint functions. Thus $Pn(Zn)$ $(n=1,2,\ldots,N\ )$ and $p_n\,(y_n)$ $(n=2,3,\ldots,N)$ is defined as follows :

$$P_n(\underline{Z}_n) = \text{OPT}\left\{F_n(f_n(x_n), f_{n-1}(x_{n-1}), \ldots, f_1(x_1); (x_n, \ldots, x_1) \in S_n(Z_n)\right\}$$

$$p_n(\underline{y}_n) = \text{OPT}\left\{f_n(x_n); \; x_n \in S_n(y_n)\right\} \tag{2.3.24}$$

The reason the objective function needs to be strongly decomposable is that if it is strongly decomposable, any improvement in $p_n\,(Y_n)$ will guarantee the solution to be as good as, by Kisashi Mine and Katsuhhisa Ohno (1970), in the last stage. Below is a theorm proved by Kisashi Mine and Katsuhhisa Ohno (1970) :

Theorem. The principal objective function is strongly decomposable with separating functions h_n (n=2,3,...,N)). Also for principal constraint functions Gm, which are strongly decomposable for m=1,...,K, weakly decomposable for m=K+1,...,L, strictly decomposable for m=L+1,...,M, with the separating functions hnm (m=1,2,...,M, n=2,3,...,N) Then the principal problem can be decomposed into subproblem and the following recursive relations hold for n=2,3,...,N.

$$P_n(\underline{Z}_n) = \mathrm{OPT}\left\{h_n(p_n(\underline{y}_n),\ P_{n-1}(\underline{Z}_{n-1}));\ \underline{y}_n \in V_n\right\}$$

The proof can be shown as follows :

$$\begin{aligned} P_n(\underline{Z}_n) &= \mathrm{OPT}\left\{F_n(f_n(x_n), f_{n-1}(x_{n-1}), \ldots, f_1(x_1)); (x_n, \ldots, x_1) \in S_n(\underline{Z}_n)\right\} \\ &= \mathrm{OPT}\left\{h_n(f_n(x_n), F_{n-1}(f_{n-1}(x_{n-1}), \ldots, f_1(x_1))\ ; \right. \\ &\qquad\qquad \left.(x_n, \ldots, x_1) \in \bigcup_{\underline{y}_n \in V_n} s(\underline{y}_n) X S_{n-1}(\underline{Z}_{n-1})\right\} \\ &= \mathrm{OPT}\left\{\mathrm{OPT}\left\{h_n(f_n(x_n), F_{n-1}(f_{n-1}(x_{n-1}), \ldots, f_1(x_1)));\right.\right. \\ &\qquad\qquad \left.(x_n, \ldots, x_1) \in \left\{s_n(\underline{y}_n) X S_{n-1}(\underline{Z}_{n-1})\right\}\right\} \\ &\qquad\qquad \left.;\ \underline{y}_n \in V_n\right\} \\ &= \mathrm{OPT}\left\{h_n(\mathrm{OPT}\left\{f_n(x_n); X_n \in S_n(\underline{y}_n)\right\}, \mathrm{OPT}\left\{F_{n-1}(f_{n-1}(x_{n-1}),\right.\right. \\ &\qquad\qquad \ldots, f_1(x_1)); (x_n, \ldots, x_1) \\ &\qquad\qquad \left.\left. S_{n-1}(\underline{Z}_{n-1});\ \underline{y}_n \in V_n\right\}\right. \\ &= \mathrm{OPT}\left\{h_n(p_n(\underline{y}_n),\ P_{n-1}(\underline{Z}_{n-1});\ \underline{y}_n \in V_n\right\} \qquad (2.3.25) \end{aligned}$$

The above theorem indicates that under decomposibility and left-continuous conditions, a principal problem can be decomposed into subproblems :

$$P_1(\underline{Z}_1),\ p_2(\underline{y}_2),\ p_3(\underline{y}_3), \ldots\ldots,\ p_N(\underline{y}_N)$$

To solve $P_1(\underline{Z}_1)$ at the first step, and $p_2(\underline{y}_2)$ at the second, combine $P_1(\underline{Z}_1)$ with $p_2(\underline{y}_2)$ by h_2 to obtain $P_2(\underline{Z}_2)$. In the third step solve $p_3(\underline{y}_3)$, and again combine $P_2(\underline{Z}_2)$ with $p_3(\underline{y}_3)$ by h_3. This procedure will continuously arrive at the final solution $P_N(Z_N)$. As an extension of the above theorem, a corollary is given as follows :

Corollary. Assuming the objective function f is weakly decomposable with separating functions h_n, then the inequality constraint functions, q_m (m<=L), are weakly decomposable and left-continuous with separating functions h_{nm}. The equality constraint functions q_m (m>L) are strictly decomposable with separating functions h_{nm}, as, for n=2,3,...,N, and real valued vectors $Z_n = (Z_{n1},\dots,Z_{nm})$

The proof is omitted due to its similarity to the separating form of the strongly decomposable problem.

Monotone property, instead of non-decreasing or increasing, can be expected to serve as a sufficient condition for decomposibility of a principal problem. The only modification to this is to select a suitable >= or <= as a proper or lower bound.

2.4 Problem formulation

Assume that a control problem of a large-scale system can be stated as follows :

$$\underline{X}(t+1) = \underline{A}(t)\underline{X}(t) + \underline{B}(t)\underline{U}(t) + \underline{C}(t)\underline{D}(t), \qquad \underline{X}(0) = \underline{X}^0 \tag{2.4.1}$$

$$\underline{Y}(t+1) = \underline{E}(t)\underline{X}(t) \tag{2.4.2}$$

Here $\underline{X}(t)$, $\underline{Y}(t)$, $\underline{U}(t)$, $\underline{D}(t)$ are state vectors, output vectors, control vectors and disturbance vectors at time t, respectively. The above problem is the most general form of a linear dynamic interconnected system with disturbance. As a matter of fact, the problem need not be linear, it can be quadratic or non-linear, as long as it satisfies the separable condition. Suppose this system is partitioned into p subsystems, each subsystem i having N state variables, and M control variables. So $\sum_{K=1}^{p} N_K = N$, and $\sum_{K=1}^{p} M_K = M$, where N,M are the total numbers of state variables and control variables of the whole system. Furthermore, in a practical stationary system, control vectors can always be assumed to be entering the ith subsystem only, and all disturbance terms as D(t) = 0. Thus, the problem becomes:

$$\underline{X}_i(t+1) = \underline{A}_{ii}(t)\underline{X}_i(t) + \sum_{j} \underline{A}_{ij}(t)\underline{X}_j(t) + \underline{B}_{ii}(t)\underline{U}_i(t)$$

$$\underline{X}_i(0) = \underline{X}_i^0 \tag{2.4.3}$$

$$\underline{Y}_i(t+1) = \underline{E}_{ii}(t)\underline{X}_i(t) \qquad i = 1,2,\ldots,p \tag{2.4.4}$$

where the subindex i stands for the number of subsystems. Therefore the whole system can be viewed as partitioned in the following way :

$$\left[\begin{array}{c} \underline{X}_1(t+1) \\ \underline{X}_2(t+1) \\ \hline \vdots \\ \hline \underline{X}_N(t+1) \end{array}\right] = \left(\begin{array}{cc|c|c} \underline{A}_{11}(t) & \underline{A}_{12}(t) & \cdots \underline{A}_{1N-1}(t) & \underline{A}_{1N}(t) \\ \underline{A}_{21}(t) & \underline{A}_{22}(t) & \cdots \underline{A}_{2N-1}(t) & \underline{A}_{2N}(t) \\ \hline & \vdots & & \vdots \\ \hline \underline{A}_{N1}(t) & \underline{A}_{N2}(t) & \cdots \underline{A}_{NN-1}(t) & \underline{A}_{NN}(t) \end{array}\right) \left[\begin{array}{c} \underline{X}_1(t) \\ \underline{X}_2(t) \\ \hline \vdots \\ \hline \underline{X}_N(t) \end{array}\right] \left(\begin{array}{cccc} \underline{B}_{11}(t) & & & \underline{0} \\ & \underline{B}_{22}(t) & & \\ \underline{0} & & \ddots & \\ & & & \underline{B}_{NN}(t) \end{array}\right) \left[\begin{array}{c} \underline{U}_1(t) \\ \hline \underline{U}_2(t) \\ \hline \vdots \\ \hline \underline{U}_N(t) \end{array}\right]$$

(2.4.5)

The dotted lines show one of the many possible ways of decomposition in this problem. Note that the decomposition of state vectors need not be consistent with control vectors.

To achieve both the ease and efficiency of SDP, it is suitable to assume the coupling terms A_{ij}'s will satisfy :

$$\sum_j \text{rank}\left\{\underline{A}_{ij}(t)\right\} < N_j, \qquad \forall\ t, i$$

This condition implies the degree of interaction, and is also a sufficient condition to get a sparse interaction matrix (to look into the degree of interaction on large-scale systems, see [Douglas, 1982]). Also, the state vectors and control vectors are constrained according to:

$$G_i(x_i(t), u_i(t)) \leq 0 \qquad \forall\ t, i$$

For the presupposition stated above, it is necessary to assume a strongly decomposable F is given, and to denote the objective function by J, thus the whole problem becomes:

$$\min J = F\,(\underline{x}_N(t), \underline{x}_{N-1}(t), \ldots, \underline{x}_1(t))$$

subject to :

$$\underline{x}_i(t+1) = \underline{A}_{ii}(t)\underline{x}_i(t) + \underline{A}_{ij}(t)\underline{x}_j(t) + \underline{B}_{ii}(t)\underline{U}_i(t)$$

$$\underline{x}_i(0) = \underline{x}_i^0 \qquad G\,(\underline{x}_i(t), \underline{U}_i(t)) \quad 0$$

$$i = 1,2,3,\ldots,p$$
$$t = 0,1,\ldots,T-1 \qquad (2.4.6)$$

where p is the total number of subsystems, and 'T' represents the final time state. Then the f_i's are separating functions of the performance function F. The control problem then becomes choosing a control and a state in the sequence $\underline{X}_i(t)$, $\underline{U}_i(t)$ so that J is minimal. Since F is assumed to be strongly decomposable, it can be represented as :

$$F(\underline{X}_N(t), \underline{X}_{N-1}(t), \ldots, \underline{X}_1(t)) = F_p(\underline{X}_p(t), \underline{X}_{p-1}(t), \ldots, \underline{X}_1(t))$$
$$= f_p(\underline{X}_p(t), f_{p-1}(\underline{X}_{p-1}(t), \ldots)\ldots)$$

Here 'p' is the number of the total subsystems.

As t and i proceed, strong decomposibility assures a same or better solution than previous time slots and stages. Defining this problem as principle problem P [Cline, 1977], it is stated as :

$$P = \min\Big\{ F_p(f_p, \ldots f_1);\ \underline{X}_i(t+1) = \underline{A}_{ii}(t)\underline{X}_i(t) + \underline{B}_{ii}(t)\underline{U}_i(t),$$
$$\underline{X}_i(0) = \underline{X}_{i,}^0 \quad (\underline{X}_i(t), \underline{U}_i(t)) \in \widetilde{\underline{Z}}_i\ ;$$
$$i = 1,2,3,\ldots,p,\ t = 0,1,2,\ldots,T-1\Big\}$$

Where $\tilde{\underline{Z}}_i$ represents the feasible set of $\underline{X}_i(t),\underline{U}_i(t)$ to satisfy the constraints : $G_i(\underline{X}_i(t),\underline{U}_i(t))<=0$ (2.4.7)

This is a typical optimization control problem defined in decomposable manner. In the following chapter, SDP is used to solve the control problem stated above.

CHAPTER 3

SPATIAL DYNAMIC PROGRAMMING ON OPTIMIZATION CONTROL PROBLEM

3.1 Algorithm of SDP

Vector $\underline{Z}_i = (\ x_i(1),\ x_i(2),\ \ldots\ ,\ x_i(T),\ u_i(1), u_i(2), \ldots, u_i(T)$, denoting the sequence of state variables and control variables in subsystem i, from time 0 to time T. Assuming $\underline{Z} = (Z_1, Z_2, \ldots, Z_N)$, $\underline{X}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$, N: no. of subsystem where the latter denotes the initial conditions. Then the equation (2.4.3), (2.4.4) can be represented as follows :

$$\begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1N} & Z_1 \\ Q_{21} & Q_{22} & \cdots & Q_{2N} & Z_2 \\ \vdots & & & \vdots & \vdots \\ Q_{N1} & Q_{N2} & \cdots & Q_{NN} & Z_N \end{bmatrix} + \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1N} & X_1^0 \\ U_{21} & U_{22} & \cdots & U_{2N} & X_2^0 \\ \vdots & \vdots & & \vdots & \vdots \\ U_{N1} & U_{N} & \cdots & U_{NN} & X_N^0 \end{bmatrix} = \underline{0} \qquad U_{ii} = \begin{bmatrix} -A_{ii}(0) \\ \\ \underline{0} \end{bmatrix}$$

(3.1.1)

Where :

$$Q_{ii} = \left[\begin{array}{cccc|cccc} 1 & & & & -B_{ii}(0) & & & \\ -A_{ii}(1) & & & & & -B_{ii}(1) & & \\ & -A_{ii}(2) & & & & & -B_{ii}(2) & \\ & & \ddots & & & & & \ddots \\ & & & -A_{ii}(T-1) & & & & -B_{ii}(T-1) \end{array}\right] \quad Q_{ij} = \left[\begin{array}{cccc|c} 0 & & & & \\ -A_{ij}(1) & & & & \\ & -A_{ij}(2) & & & \underline{0} \\ & & \ddots & & \\ & & & -A_{ij}(T-1) & \end{array}\right] \quad U_{ij} = \begin{bmatrix} -A_{ij}(0) \\ \\ \underline{0} \end{bmatrix}$$

Then define S as :

$$S_i = \sum_{j=1}^{N} Q_{ij} Z_j \ + \ U_{ij} X_j^0 \qquad i = 1, 2, \ldots, N \tag{3.1.2}$$

When S_i represents all the constraints within subsystem i, Qij's are the coupling factors, and U_{ij}'s are terms related to initial conditions. This equation covers the time span from t = 0,..., T-1. From (3.1.1) and (3.1.2), it is easy to show that:

$$S = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ \vdots \\ S_N \end{bmatrix} = \underline{0} \qquad (3.1.3)$$

Since Si is a function of $(Z_1, Z_2, \ldots, Z_N)$, it is necessary to define function gi . Here q_i's are functions of Zi only.

The equation (3.1.3) represents the constraints of whole system, which is shown above to be $S(\underline{q}_1(\underline{Z}_1), \underline{q}_2(\underline{Z}_2), \ldots, g_N(Z_N)) = \underline{0}$. This result implies that the overall constraints in the systems can be partitioned into many interconnected sets of constraints in subsystems. Because of the additivity of coupling terms, S is separable and strongly decomposable with the separating functions hi(x,y) = x + y, i = 1, 2,..., N. In it's decomposable form, S can be shown as follows :

$$\begin{aligned} S &= (g_1(\underline{Z}_1), g_2(\underline{Z}_2), \ldots, g_N(\underline{Z}_N)) \\ &= h_1(g_1(\underline{Z}_1), h_2(g_2(\underline{Z}_2), \ldots, h_{N-1}(g_{N-1}(\underline{Z}_{N-1})+g_N(\underline{Z}_N)) \\ &= h_N(g_N(\underline{Z}_N, \ldots, h_2 (g_2(\underline{Z}_2), h_1(\underline{Z}_1)) \end{aligned} \qquad (3.1.4)$$

Then the principle problem (2.3.7) becomes :

$$P = \min \left\{ F(f_N, f_{N-1}, \ldots, f_2, f_1);\ S(h_N, h_{N-1}, \ldots, h_1) = 0,\ \underline{Z}_i \in \underline{\widetilde{Z}}_i \right\}$$
$$= \min \left\{ F(p_N(\underline{Z}_N), p_{N-1}(\underline{Z}_{N-1}), \ldots, p_1(\underline{Z}_1); S(g_N(\underline{Z}_N), \ldots, g_1(\underline{Z}_1) = 0,\ \underline{Z}_i \in \underline{\widetilde{Z}}_i \right\}$$

(3.1.5)

Both the performance functions and constraint functions are in their decomposable forms now. Those pi's, i = 1, 2,..., N, are the reall-valued functions associated with variables in every subsystem (under the assumption that objective function is discomposable with the separating function fi). The control problem may be restated as discovering the control sequence : $\{\underline{Z}_1, \underline{Z}_2, \ldots, \underline{Z}_N\}$

such that min F, and will satisfy S=0, $\underline{Z}_i \in \underline{\widetilde{Z}}_i$. The problem formulation is summarized in the following :

$$\underline{\Phi}_i - \underline{\Phi}_{i-1} = g_i(\underline{Z}_i) = \begin{bmatrix} \underline{Q}_{1i}\underline{Z}_i & + & \underline{U}_{1i}\underline{X}_i^0 \\ \underline{Q}_{2i}\underline{Z}_i & + & \underline{U}_{2i}\underline{X}_i^0 \\ \vdots & & \vdots \\ \underline{Q}_{Ni}\underline{Z}_i & + & \underline{U}_{Ni}\underline{X}_i^0 \end{bmatrix}$$

$$\underline{\Phi}_i = g_i(\underline{Z}_i) + \underline{\Phi}_{i-1} \qquad i = 1,2,\ldots,N$$

(3.1.6)

The recurrence relationship of the priincipal problem can be divided into two parts: one for the performance criteria function, the other for the constraints. For the objective function F, The recurrence formula of SDP is as follows:

$$P_i(\underline{\Phi}_i) = \min_{\underline{\Phi}_i} \left\{ f_i(p_i(\underline{Z}_i), P_{i-1}(\underline{\Phi}_{i-1})); hi(g_i(\underline{Z}_i), h_{i-1}(\underline{\Phi}_{i-1})), \underline{Z}_i \in \underline{\widetilde{Z}}_i, i = 1,2,\ldots, N \right\}$$

$P_N(\underline{\Phi}_N) = P_N(\underline{0})$ is the original problem

(3.1.7)

The above result has been obtained as a theorem in Cline's paper [Cline, 1977].

3.2 System aspects of SDP

The conditions required to implement the SDP are weak. Only the decomposability has to be assured for both objective function and constraints. As a matter of fact, any arbitrary constraint set is sequential separable [Chong, 1978]. It can be shown that the separability and monotonicity of the objective function constitutes the sufficient condition for the application of dynamic programming.

Also, a global optimality theorem for SDP has been proposed in 1978 [McEntire, 1978], which states that only a weak form function is necessary to guarantee the global optimality of the final solution. This is the extended result [Cline, 1977], in which the interaction term is assumed to be additive. In the past, dynamic programming has been thought to be useful on systems which do not process a complicated feedback structure, or an intervined network structure. That is, only systems with acyclic structure would be good candidates for dynamic programming [Aris, 1964]. Some complex processes in chemical plants with recycle streams, for example, seemed unsuitable to the application of dynamic programming at that time [Lee, 1967], because of 'dimensionality difficulty'. These problems may become tractable by means of spatial dynamic programming, because the 'recycled stream' can be treated as an interaction with other subsystems. Actually, SDP is a combining technique for decomposition and coordination. Both of these (decomposition and coordination) are well-developed techniques. SDP decomposes the overall large-scale system into a sequence of subsystems, applying dynamic programming over each system. Each time this is to obtain a suboptimal strategy rather than a global optimal strategy, and suitable coordination variables must be used to compensate for interactions among the subsystems. When all the subsystems have been included, the global optimum will be obtained, with the ease and efficiency of SDP residing in the decomposition of whole system into smaller subsystems. Ideally, the SDP technique can deal with the most general structure form of the large-scale system with the following configuration :

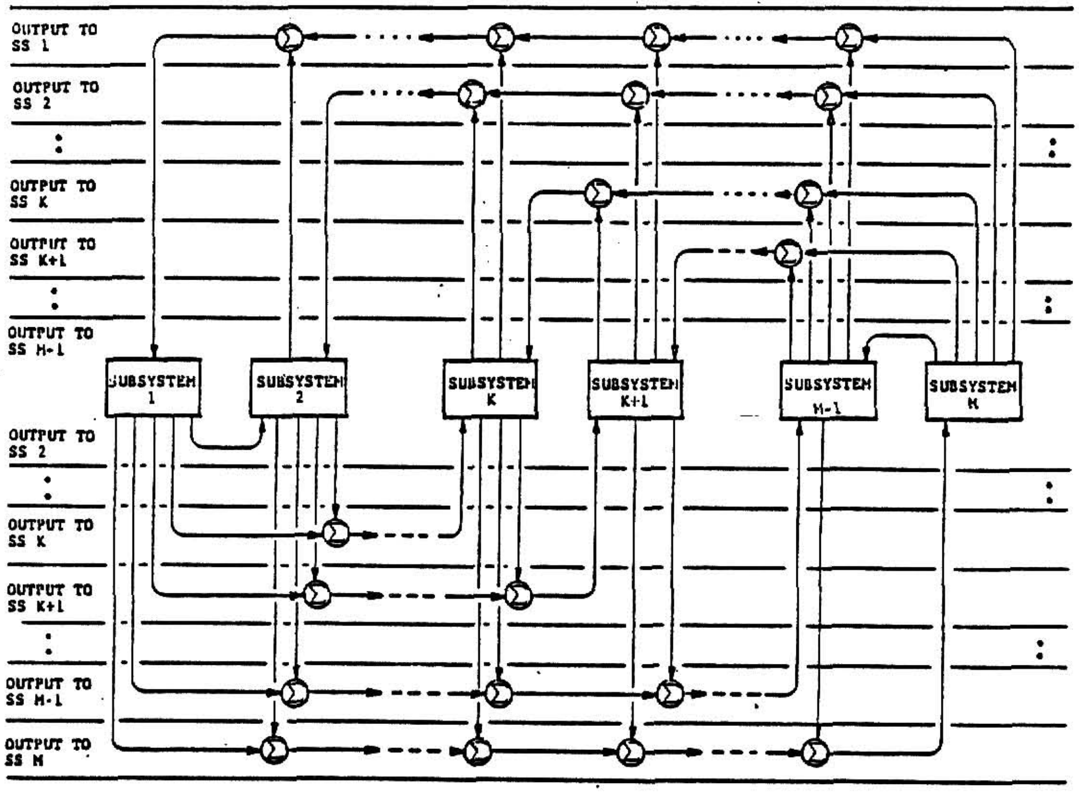


Fig. 3.2.1 Large-scale System with M Subsystems

Operator $\oplus$ is used for additive constraints, which can be replaced by other appropriate operators depending on the separating function. The connection link in this figure stands for the interacting terms between subsystems. Only when these coupling terms are sparse can SDP show its promising efficiency. By means of SDP, the optimal control of the total system can be implemented by a series of local controllers, one for each subsystem (which means each subsystem has its own set of control variables.)
Each subsystem then communicates with its interacting subsystems by state variables. Thus, for a stationary system, the two-level method is appropriate, as follows :

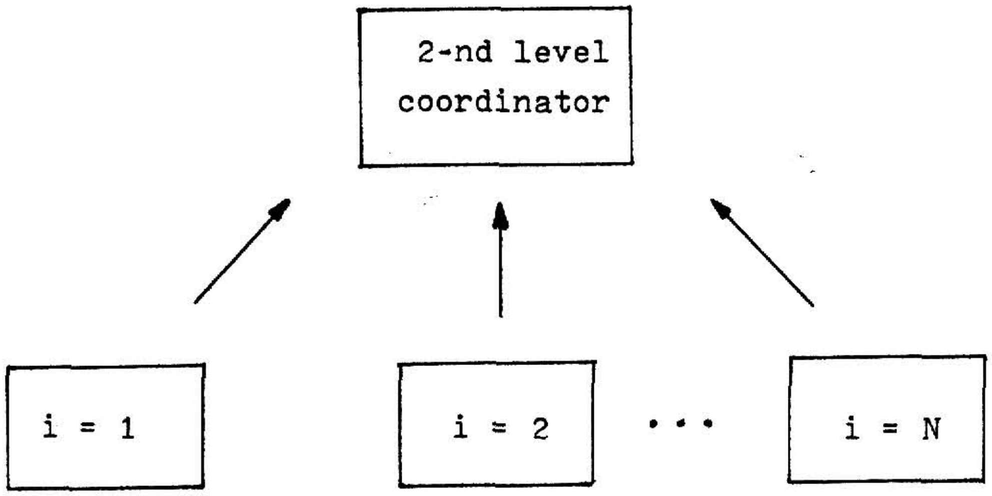


Fig. 3.2.2 Two level method for stationary interacting system with N subsystems

But for a dynamic system, an additional level should be included to account for indices over the time span. The idea in the dynamic system is essentially equivalent to the

'Three level method of Tamura' [Singh, 1980,1], [Singh, 1980,2], [Tamura, 1973]. The optimation structure is as follows:

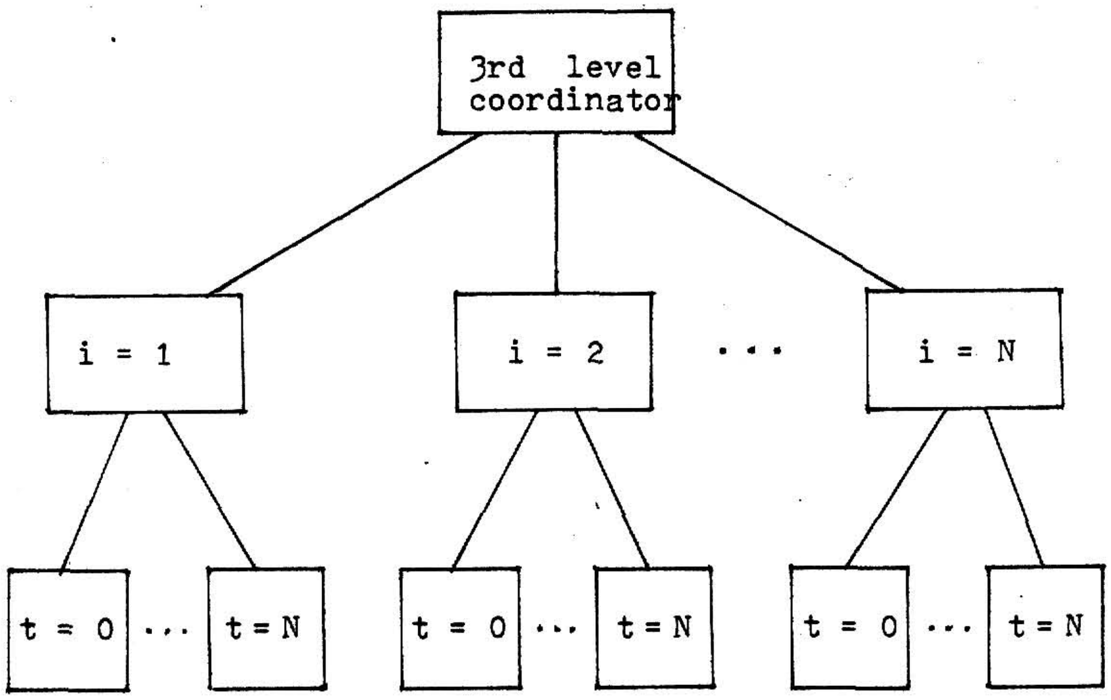


Fig. 3.2.3 Three level method for dynamical interacting system with N subsystems

The lowest level in the above figure accounts for the time span of the dynamic system, the second level takes care of the decomposition algorithm, and the top level serves as a monitoring coordinator to supervise the interaction between subsystems.

Given a set of possible values for an interaction variable, applied SDP will optimize the current subsystem. If a closed-form solution can be obtained for this parameterized interaction, then the computation tasks would be reduced a great deal. An example is the derivation of the Riccati equation for the LQ problem using the principle of optimality and dynamic programming [Li, 1983]. Furthermore, if the system can be assumed to be in a steady-state, the design methodology becomes suboptimal but much more computationally attractive [Li, 1982]. Even if a closed-form solution may not be possible, computational approaches provide attractive alternatives, such as dynamic programming or the successive approximation technique [Bellman, 1962]. Bellman's technique is fairly easy to implement. However, the problem is that the global optimality can only be guaranteed on some conditions, although its convergence has been proved by R. E. Larson and A. J. Korsak in [Larson, 1970], [Korsak, 1970]. Another promising feature of SDP is the description of the state variable, which may not be easily identified for some optimization problems. In these situations, SDP combines with descriptor variable theory to solve the large-scale system problem [Larson, 1978].

3.3 Advantages and drawback of SDP

According to some control scientists' opinion, dynamic programming is not powerful enough to serve as an unified 'structure' in solving optimal control problems of the large-scale system [Sandell, 1978]. But, as the computer technology evolves, capacities and speeds increase with the performance/cost ratio droping drastically. This trend gives a new perspective for the application of SDP on large-scale systems. Besides, the other advantages of SDP can be summarized as follows:

a) Local controllers evaluate alternatives independently and paralelly with other local controllers. A central coordinator selects the optimal policy based on the information given by local controller one at a time. This kind of structure greatly reduces the 'dimensionality difficulty'.

b) No particular form is required for the subsystem description. Which can be linear, quadratic, non-linear, or even descriptor form. This feature expands the potential application field without model simplification.

c) The purturbance can be viewed as interaction terms, which are neither input variables nor output variables. This feature makes SDP applicable on some special systems, such as the economic systems. This way, variables may affect many subsystems without itself being an input or output variable of any subsystem.

d) Reconfiguration capability is excellent for SDP implementation. In case of any failure of a subsystem, the interaction as well as the sequence order can be adjusted to reflect the true situation. This characteristic is very important for defense purposes.

e) Since SDP treats interaction variables satisfactorily, the non-serial system can also be dealt with with great ease. This property enables SDP to be applied upon non-serial systems such as river-pollution problems.

In spite of the advantages of SDP stated above, there are also some drawbacks :

a) The efficiency of SDP depends heavily on the grouping and sequencing of the variables, of which there is no standard rule for effective decomposition and sequence ordering. Only a few basic principles have been proposed by [Larson, 1979]. Two of these are, 'If one subsystem affects a second subsystem but not vice-versa, the latter subsystem should proceed the former in the sequence.', and 'When the interaction is sparse, then the decomposition is more effective.'.

b) Computational experiences from the implementation of SDP show that it is very important to choose variables with great care so as to avoid the 'curse of dimensionality'. This drawback is similar to classical dynamic programming.

3.4 Application

A water-treatment problem of 7-reach river is presented and solved by spatial dynamic programming [Chapter 4], which adopts the model from Camp, [1963], and Dobbins, [1964]. Another application of SDP on the same type of river pollution control problem, but with a different model [Kendric, 1970], was modified by Tamura [Tamura, 1975], and has been worked out by Li [Li, 1982] with steady-state approximations.

There are some other fields of the large-scale system which have been explored by SDP. They are:

1) Ballistic missile defense area [Anton, 1978]
2) Electric energy systems area [Stengel, 1979]
3) Control of communication networks
4) Vehicle dispatching for transportation systems
5) Integrated control of multiple-process industrial plants

The initial results from these areas are fairly successful [Larson, 1979].

Essentially, the SDP structure is well suited for decentralized control and distributed data processing. The only question remaining is to find out formal decomposition rules and an efficient procedure for enumerating feasible sets of interactions. This leads to the open study field of SDP which still needs further exploration [McEntire, 1978].

CHAPTER 4

APPLICATION OF SDP ON WATER TREATMENT PROBLEM

4.1 Introduction

The topic of water quality control has received wide attention for the past twenty years. Many optimizing techniques, which are shown in the following figure, have been developed to solve the problems in this category.

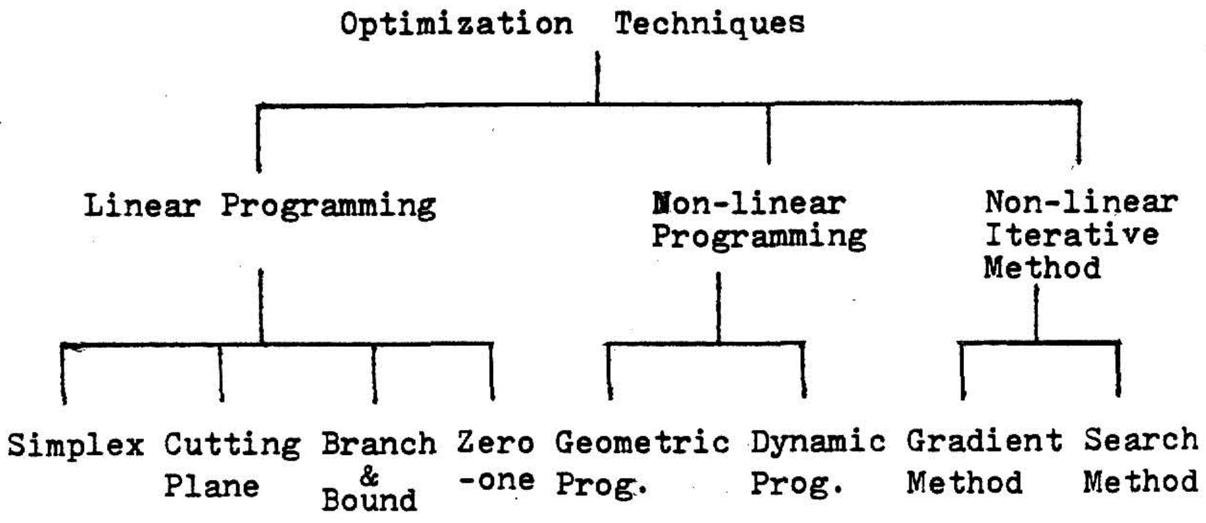


Fig. 4.1.1 Classification of optimization techniques

Although these techniques have been implemented successfully under their specific enviroments, they do have their own drawbacks. For example, the linear programming technique can only be applied to those problems with linear structure. The gradient method suffers from the uncertainty of the truely global optimum. Also, the dynamic programming technique is prohibited when the non-serial structured problem is of concern. All the other optimization techniques, such as the geometric programming technique, the search method, etc., may require that some specific structures of the problem itself are assumed. Otherwise the optimum can not be guaranteed or the computational tasks will be too involved to implement. These shortcomings limit the prevailing uses of each optimization technique in its specific environment.

In fact, the dynamic programming technique can be modified to solve non-serial structured problems. The basic idea of this modification is to treat each interaction term between subsystems as an individual state variable, since each possible value of the interaction corresponds to a specific state in the system. Then the objective function is optimized due to all feasible states by selecting the best control policy at each stage. Proceeding stage by stage, this method will obtain global optimum after all the sbsystems have been considered and all the possible values of interaction have been examined. The only disadvantage here is the increasing number of state variables. This increase makes the complexity of the problem worse, and may involve the so called 'dimensionality difficulty' of dynamic programming. Therefore the problem may become too tedious or even impossible to solve because of the involved computational tasks.

The SDP technique introduced in the previous two chapters are capable of dealing with the water treatment problems, which are serial-structured in nature, in a more efficint way without involving too much of the dimensionality difficulty . The basic idea of converting the interaction terms into state variables is the same as that stated in the last paragraph. The only difference is that here the problem is optimized due to summation of the possible values of all interaction variables instead of according to each specific value in the individual state variable. This feature gives the SDP technique the capability of overcoming dimensionality difficulty . The SDP technique on water treatment problem will be described in detail in the following sections.

4.2 An example in water treatment problem

To illustrate the SDP approach, a hypothetical river system solved by Lee [Lee, 1972] is now solved by the spatial dynamic programming method. The system is shown schematically in the following figure (Fig. 4.2.1)

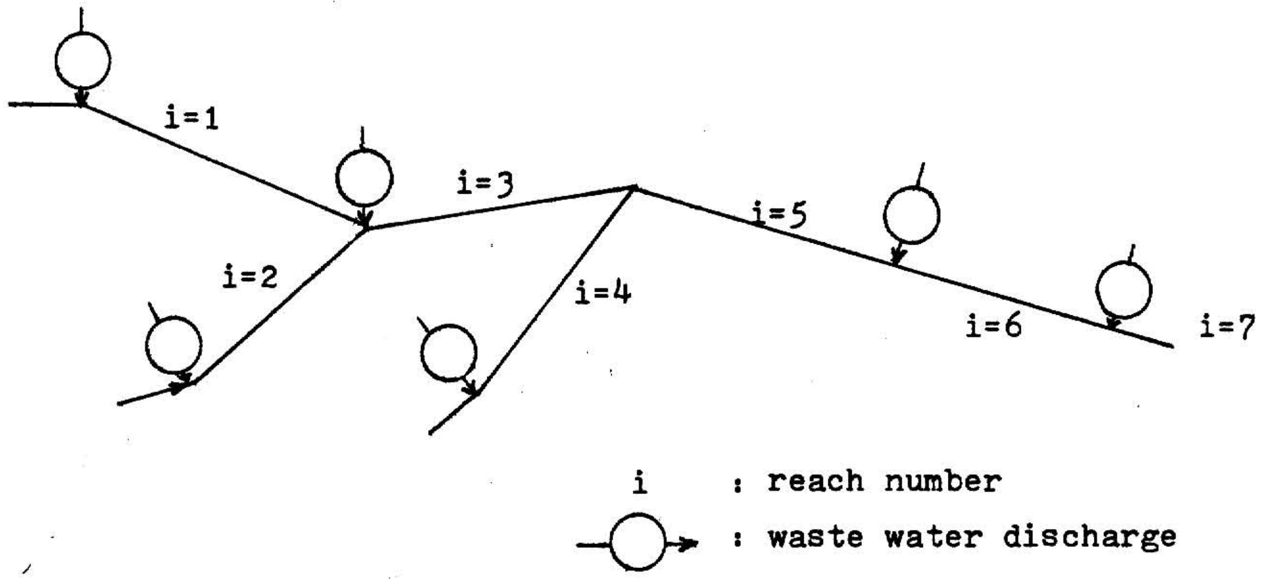


Fig 4.2.1 Schematic representation of a river system

There are six waste water discharge streams and seven reaches. Each reach may or may not have a tributary or a waste water discharge. For example, reach five does not have waste water discharge , while reach six does not have a tributary. It is assumed that waste water treatment facilities exist and are currently removing a sufficient amount of BOD to satisfy the stream quality standard. However, as time proceeds, one can anticipate that the BOD will increase considerably according to the degree of industrial pollution. Thus, it is necessary to do additional treatment at a minimum cost.

The same river system data used by Lee [Lee, 1972] , are used here and are summarized in the following table (Table

4.2.1).

Table 4.2.1 System parameters and data

Reach	1	2	3	4	5	6	7
t_i (Days)	.235	1.330	1.087	2.067	.306	1.050	6.130
W_i (MGD)	5	37	8	14	---	26	41
T_i (MGD)	1355	1290	1360	296	310	---	---
Q_i (MGD)	1360	1327	2695	310	3005	3031	3072
A_{si} (mg/l)	10.20	9.95	9.00	9.54	9.00	8.35	8.17
D_i^{max} (mg/l)	3.20	2.45	2.00	3.54	2.50	2.35	4.17
A_{Wi} (mg/l)	1.0	1.0	1.0	1.0	---	1.0	1.0
A (mg/l)	9.50	8.00	---	9.70	---	---	---
B (mg/l)	1.66	0.68	---	1.0	---	---	---
K_1 (days^{-1})	.31	.41	.36	.35	.34	.35	.30
K_2 (days^{-1})	1.02	.60	.63	.09	.72	.14	.02
K_3 (days^{-1})	.02	.03	.04	.04	.05	.06	.00
ε (mg/l/ day)	.85	.14	.18	.05	.39	.07	.00
η (mg/l/ day)	.15	.14	.14	.11	.11	.13	.00
1990 BOD	248	408	240	1440	---	2180	279
Removal	.67	.10	.26	.24	---	.12	.26

These data include the flow data (Qi,Ti,Wi), the BOD and DO concentrations, (BWi,BTi,AWi,Ai), parameters such as reach residence time length (ti), quality standard, (Dimax), and other relevant constants(K_{i1}, K_{i2}, K_{i3}, ξ, η). The coffecients of quadratic cost function are also listed in the above table. Three cofficients (ai,bi,ci) are obtained by a typical quadratic fit of cost curve which is assumed to be quadratic [Lee, 1972]. This non-linear approach is different from Loucks' [Loucks, 1967], in that the linear approach has been used to describe the cost function, presenting a more realistic representation of the cost criterion.

The mathematic model used here, which governs the dynamical characteristic transition of system variables, is adopted from Camp [Camp, 1963], which can be represented in the following form :

$$B(t) = (B(0) - \frac{\eta}{K_1+K_3})\exp(-(K_1+K_3)t) + \frac{\eta}{K_1+K_3} \qquad (4.2.1)$$

$$D(t) = \frac{K_1}{K_2 - (K_1 + K_3)}(B(0) - \frac{\eta}{K_1+K_3}) \cdot (\exp((-K_1 - K_3)t) - \exp(-K_2 t)) + \frac{K_1}{K_2}(\frac{\eta}{K_1+K_3} - \frac{\xi}{K_1})(1 - \exp(-K_2 t)) + D(0)\exp(-K_2 t) \qquad (4.2.2)$$

B(t), and D(t) represent the BOD concentration and the oxygen deficit at t=t along the reach. 't' is the point at which t length (residence time length) downstream from the initial point t=0. K1 and K3 represent the deoxygenation and sedimentation rate. K2i is the reaeration rate of DO returnning to the stream. γ represents the BOD addition rate due to run-off and scour along the stream. ξ represents the rate of oxygen production or reduction due to plant photosynthesis and respiration. The simpler Streeter-Phelps saq equations can be obtained by assuming γ, ξ, K_3 are equal to zero in equations, (4.2.1) and (4.2.2). The resulting simpler equations are as follows :

$$B(t) = B(0)\exp(-K_1 t) \tag{4.2.3}$$

$$D(t) = \frac{K_1}{K_2-K_1} B(0)(\exp(-K_1 t)-\exp(-K_2 t))+D(0)\exp(-K_2 t) \tag{4.2.4}$$

These two equations stated above are identical to the 'transformation equation', the name used in classical dynamic programming method. These equations govern the dynamical feature of the system. Besides these two equations, there are other equality constraints and inequality constraints which are encountered in the water treatment problem. By assuming complete mixing of the fluid flow at all points where a tributary or a waste water effluent enters a reach or stream, and applying the material balance, the equality constraints can be obtained as follows

$$Q_i = Q_{i-1} + T_i + W_i \tag{4.2.5}$$

Qi : total flow quantity at reach i

Ti : tributary flow quantity at reach i

Wi : discharge waste water flow quantity at station i

$$B_i(0) = \frac{Q_{i-1}B_{i-1}(f) + T_iB_{Ti} + W_iB_{Wi}}{Q_i} \tag{4.2.6}$$

Bi(0) : initial BOD concentration at reach i

Bi(f) : BOD concentration at the end of reach i

BWi: BOD concentration of discharge waste water from station i

$$A_i(0) = \frac{Q_{i-1}A_{i-1}(f) + T_iA_{Ti} + W_iA_{Wi}}{Q_i} \tag{4.2.7}$$

Ai(0) : DO concentration at the beginning of reach i

Ai(f) : DO concentration at the end of reach i

AWi : DO concentration of wastewater from station i

$$A_i(0) = A_{si} - D_i(0) \tag{4.2.8}$$

Asi : represents the DO satuation concentration at reach i

Di(0) : DO deficit at the beginningof reach i

The above equations (4.2.5 - 4.2.8) constitute the equality constrants of the water treatment problem concerned. Moreover, the DO deficit, Di(t), at each point along reach i, must not exceed the allowable maximum in that reach, that is :

$$D_i(t) \leqq D_i^{max} \tag{4.2.9}$$

0 <= t <= ti

ti : represents the total residence time of reach i

Because of the assumption that all treatment facilities, a minimum removal of 35% is imposed to insure the absence of floating solids in the stream. Also, the maximum treatment allowed is 90% due to practical equipment limitations, this fact constitutes the second inequality constraint :

$$\underset{(35\%)}{P_i^{min}} \leqq P_i \leqq \underset{(90\%)}{P_i^{max}} \tag{4.2. 10}$$

Pi : water treatement degree at station i

P^{mm}: practical lower bound of treatment degree of water treatment

P^{max} : practical upper bound of treatment degree

The last inequality constraint is shown as :

$$0 \leqq B_{Wi} \leqq B_{Wi}^{max} \tag{4.2. 11}$$

This constraint, representing BOD as the concentration of discharged wastewater, can not exceed the total BOD concentration available for release at the station.

Finally, an objective function needs to be defined to complete the formulation of an optimization problem. In this problem, the objective is to minimize the total treatment cost due to the different degrees of treatment at all stations. That is :

$$\min \psi = \min_{p_1, p_2, \ldots, p_7} \left(\sum_{\substack{i=1 \\ i \neq 5}}^{7} C_i(p_i) \right)$$

$$= \min_{\substack{p_i \\ i \neq 5}} \sum_{i} (a_i + b_i + c_i p_i^2)$$

(4.2.12)

ai,bi,ci: coefficients of quadratic cost function

Putting together the equality constraints, the inequality constraints, and also the objective functiuon, the water treatment problem can be summarized as the following :

$$\min \psi = \min \sum_{\substack{i=1 \\ i \neq 5}}^{7} (a_i + b_i p_i + c_i p_i^2)$$

subject to :

$$B_i(0) = \frac{Q_{i-1} B_{i-1}(t_i) + T_i B_{Ti} + W_i B_i (1-p_i)}{Q_i}$$

$$A_i(0) = \frac{Q_{i-1} A_{i-1}(t_i) + T_i A_{Ti} + W_i A_i (1-p_i)}{Q_i}$$

$$D_i(0) = A_{si} - A_i(0)$$
$$B_i(t^i) = B_i(0)\exp(-K_{i1}.t^i)$$
$$D_i(t^i) = (K_{i1}/(K_{i2} - K_{i1}))B_i(0)(\exp(-K_{i1}t^i)-\exp(-K_{i2}t)$$
$$+ D_i(0)\exp(-K_{i2}t^i)$$
$$0.35 \leq p_i \leq 0.90$$
$$D_i(t^i) = D_i^{max}$$
$$i = 1,2,3,\ldots,7$$
$$t^i = 0,\Delta t_i,2\Delta t_i,\ldots,t_i-\Delta t_i,t_i \qquad (4.2.13)$$

Since the objective function and transformation equations (4.2.13) are both non-linear, the above problem is in fact a non-linear programming problem. It can not be solved by the linear programming technique. In this work the problem is solved by the spatial dynamic programming technique; which is, in fact, a modification of the traditional dynamic programming method with a different decomposition algorithm to divide the overall system into smaller subsystems.

After a brief discussion of the drawbacks of straight forward dynamic programming as applied on water treatment problems, the SDP algorithms and its application procedure will be explained in detail in the following sections.

4.3 The difficulties with straightforward dynamic programming

The most obvious difficulty in applying a straight forward dynamic programming method on water treatment problems is that : the dynamic programming technique has its limitations applied only to serial-structured problems, while this problem itself has a non-serial structure. If the river system does not not have too many tributaries and branches, the classical dynamic programming technique may be suitable in this application.

Assuming that non-serial structure has been converted to serial structure, some additional state variables needed to be created to represent the interaction terms between subsystems. Next, the straight-forward dynamic programming technique is applied to optimize the stages according to the specific value of each individual state variable. The profit gained from this conversion may not be justified because the price paid may expose the problem to the so called dimensionality difficulity of dynamic programming. Thus, the problem becomes intractable because of the computational complexity. It will be shown in the following two sections how the SDP technique is revealed through its organized decomposition algorithm by dividing the whole system to smaller ones.

4.4 SDP algorithms

To solve the above problem by SDP methods, the first step is to decompose the overall system into a sequence of small subsystems. The ease and efficiency may depend on the method of decomposition. However, as stated in the last two chapters, no existing algorithm will guarantee the best decomposition. The decomposition algorithm used here is the same as the reach number proceding it. That is, the whole hypothetical river system is decomposed into seven subsystems, with each subsystem containing a water treatment station and a reach. The exception is the fifth subsystem, which has no water treatment station at all. However, for the homogeneity of the procedure, a null water treatment station is assumed to exist with the only possible treatment degree being equal to zero. With this decomposition approach, the river system can be decompose as the following figure (Fig. 4.4.1) :

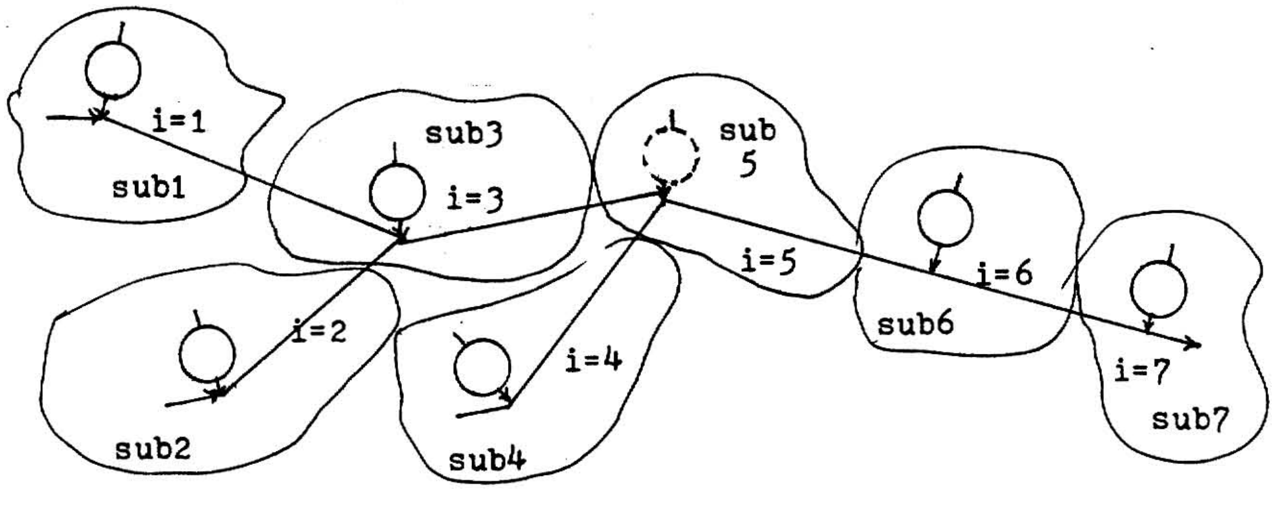


Fig. 4.4.1 Schematic decompostion of the hypothetical river system

The coupling variables and the system parameters can also be represented in the following block diagram. (Fig. 4.4.2)

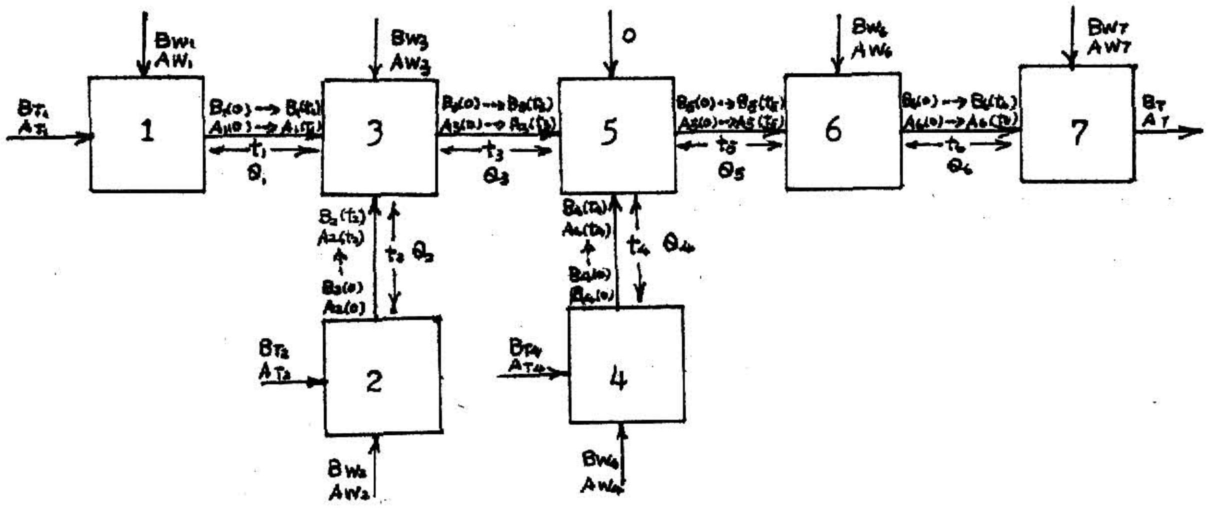
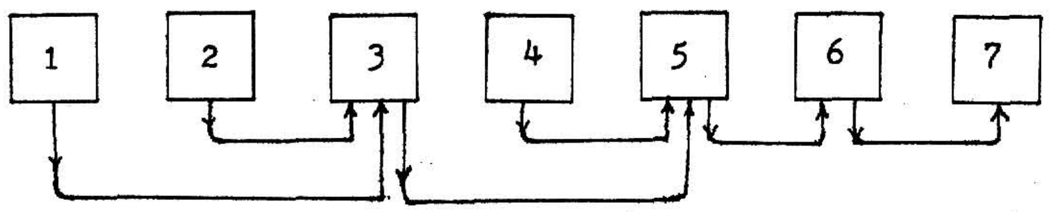


Fig. 4.4.2 Block diagram of system

Although the overall system has been decomposed into seven subsystems, it is still a non-serial structured problem. Therefore, the straightforward dynamic programming can not be applied unless a conversion from the non-serial structure to a serial structure be imposed.

By introducing the set of interaction variables A_{ij}, which represent the interaction from subsystem j to subsystem i, the non-serial problem can be converted to a serial structure problem. This is shown in the following (Fig. 4.4.3)



$\longrightarrow$: interaction exists

Fig. 4.4.3 Interaction among the sequence of subsystem

Now the whole system has been rearranged into the desired serial structure. This problem is a simple one because of its weak interaction. For the most general form of the interacting structure, each subsystem should have interaction terms from, and to, all the other systems. However, this is not the case here. In this problem, Bi(t), and Ai(t), are treated as state variables; whereas, Pi is the decision variables, while BTi, and ATi, if they exist, are treated as initial conditions. This is because the state of the current reach is affected by the terminal state of coupling reaches. For example, the starting state of reach 3 is influenced by the terminal state of reach 1 and 2; therefore, states at all time instants should be included as state vectors rather than simply state variables. By defining the state vectors $\underline{Z}i$ = (Bi(0), Bi(Δti),..., Bi(ti), Ai(0), Ai(Δti),..., Ai(ti), pi$)^T$ and $\underline{Z}_i^0$ = (BT_i, AT_i), the system constraints can be represented as the following:

$$\begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{17} \\ Q_{21} & \cdot & \cdot & Q_{27} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ Q_{61} & \cdot & \cdot & Q_{67} \\ Q_{71} & \cdot & \cdot & Q_{77} \end{bmatrix} \begin{bmatrix} \underline{Z}_1 \\ \underline{Z}_2 \\ \cdot \\ \cdot \\ \underline{Z}_6 \\ \underline{Z}_7 \end{bmatrix} + \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{17} \\ U_{21} & \cdot & \cdot & U_{27} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ U_{61} & \cdot & \cdot & U_{67} \\ U_{71} & \cdot & \cdot & U_{77} \end{bmatrix} \begin{bmatrix} \underline{Z}_1^0 \\ \underline{Z}_2^0 \\ \cdot \\ \cdot \\ \underline{Z}_6^0 \\ \underline{Z}_7^0 \end{bmatrix} = \underline{0}$$

where :

$$Q_{ii} = \left[\begin{array}{ccc:ccc:c} Q_{i_0} & & & & & & Q_{Wi}B_{Wi} \\ & \ddots & & & \underline{0} & & \\ & & 0 & & & & \\ \hdashline & & & Q_{i_0} & & & \underline{0} \\ & \underline{0} & & & \ddots & & \\ & & & & & 0 & \end{array}\right] \quad U_{ii} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -Q_{Ti} \\ 0 \\ \vdots \\ -Q_{Ti} \\ -B_{Wi}Q_{Wi} \end{bmatrix} \quad \underline{Z}_i^0 = \begin{bmatrix} Q \\ 0 \\ \vdots \\ B_{Ti} \\ 0 \\ \vdots \\ A_{Ti} \\ 1 \end{bmatrix}$$

$$Q_{ij} = \left[\begin{array}{ccc:ccc:c} 0_0 & & & & & & \\ & \ddots & & & \underline{0} & & \\ & & Q_j & & & & \\ \hdashline & & & 0 & & & \underline{0} \\ & \underline{0} & & & \ddots & & \\ & & & & & Q_j & \end{array}\right] \quad U_{ij} = \begin{bmatrix} \\ \underline{0} \\ \\ \end{bmatrix} \qquad (4.4.1)$$

The matrix form of eqn.(4.4.1) is: $\underline{Q}\underline{Z} + \underline{U}\underline{Z}^{\circ} = 0$. Here $\underline{Q}$ stands for the interaction matrix of state, $\underline{Z}$ is the state matrix, $\underline{U}$ is the interaction matrix of initial conditions, and $\underline{Z}^{o}$ represents the initial condition matrix. The $\underline{Q}$, $\underline{Z}$, $\underline{U}$ and $\underline{Z}^{o}$ are sparse matrices, that is, many elements in the matrix are 0's, making it possible to solve the problem with the SDP while avoiding the dimensionality difficulty . In order to apply SDP in the same manner as decomposition, another vector Si must be defined . This is so that all the relevant constraints are contained with respect to the subsystem. Consequently, Si is defined to be:

$$S_i = \sum_{j=1}^{7} Q_{ji}\underline{Z}_i + U_{ji}\underline{Z}_i^0 \qquad i=1,2,3,\dots,7$$

(4.4.2)

According this definition, equation (4.4.1) can be represented as:

$$\begin{aligned} S &= S_1 + S_2 + \dots + S_7 \\ &= \sum_{i=1}^{7} \Big(\sum_{i=1}^{7} Q_{ji}\underline{Z}_i + U_{ji}\underline{Z}_1^0 \Big) \\ &= \underline{Q}\ \underline{Z} + \underline{U}\ \underline{Z}^0 = \underline{0} \end{aligned}$$

(4.4.3)

Now, both the objective function $\psi = \sum_i Ci(pi)$ (4.2.12), and the constraints (4.4.3), are in their decomposable forms. Because they are added to satisfy the weakly decomposability conditions, SDP can be applied. Hence the recurrence relation is formulated as follows :

$$J_{K+1} = \min_{\xi_K} \left((a_K + b_K p_k + c_K p_k^2) + J_{K-1}(\xi_K - S_K) \right)$$

$$\xi_{K+1} = S_K + \xi_K$$

(4.4.3)

Here Sk is the interaction from, or to, subsystem k, while Di's are the summation of the interaction from subsystem 1 to subsystem k. Based on the recurrence formula (4.4.3), this water treatment problem is readily optimized as the classical dynamic programming through the stage. The detailed application procedure of SDP will be given in the next section.

4.5 Application procedure

In summarizing the problem formulation from the last few sections, and by making the objective function and constraints in recursive forms, the whole problem becomes :

$$P_i(\xi_i) = \min_{\xi_i} J_i = \Big\{ f_i(p_i(\underline{Z}_i), P_{i-1}(\xi_{i-1}) ; \\ \xi_i = h_i(S_i(\underline{Z}_i), h_{i-1}(\xi_{i-1})), \\ \underline{Z}_i \in \underline{\widetilde{Z}}_i, i=1,2,3,\ldots,7 \Big\} \qquad (4.5.1)$$

Here, $P_7(0)$ represents the original problem; while the fi's and hi's are separating functions of the objective function and constraints at subsystem i.

Because the transformation equation (4.2.13) of Di is in a rather 'complex' exponential form, the numerical solution approach is used instead of the closed form solution approach. The basic theme for the numerical solution of SDP is to obtain suboptimal policy for each given possible value of an interaction. Thus, by combining the previous suboptimal policy, with the current one, the current suboptimal policy can be obtained. After all subsystems have been included, and every possible value has been examined, the optimal policy will be achieved.

Here, in the following, three subproblems at different stages are given to illustrate the application procedure of SDP:

$$J_1 = \min_{\xi_1} \; (a_1 + b_1 p_1 + c_1 p_1^2)$$

$$S_1 = \sum_{j=1}^{7} Q_{j1}\underline{Z}_1 + U_{j1}\underline{Z}_1^0 = 1$$

$$\underline{Z}_1 \in \widetilde{\underline{Z}}_1$$

(4.5.2)

In substituting the numerical figures, the subproblem at stage 1 becomes:

$$1 = \begin{bmatrix} Q_{11} \\ Q_{21} \\ \vdots \\ Q_{71} \end{bmatrix} \cdot \underline{Z}_1 + \begin{bmatrix} U_{11} \\ U_{21} \\ \vdots \\ U_{71} \end{bmatrix} \cdot \underline{Z}_1^0 = \begin{bmatrix} Q_{11}\underline{Z}_1 + U_{11}\underline{Z}_1^0 \\ Q21\underline{Z}_1 + U_{21}\underline{Z}_1^0 \\ \vdots \\ Q_{71}\underline{Z}_1 + U_{71}\underline{Z}_1^0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2.56566-.911765p_1 \\ 2.38540-.847700p_1 \\ 9.46875 \\ 0.94626-.274698p_1 \\ 0 \end{bmatrix}$$

(4.5.3)

To find the possible value of ξ_1 is the equivalent of searching for the possible value of p1. To do this J_1 must be found, which is suboptimal at stage 1. The feasible value of p_1 means the value of $D_1(t^1)$ will not exceed the maximum allowable value, D_1max, after a particular value of p_1 has been chosen. Therefore, the lower bound, as well as upper bound, have to be determined according to the range of p_1. Then, the suboptimal policy of p is found, given that the specific value of $\hat{\xi}_1$ is between its lower and upper bound. A sequence of $p_1(\xi_1)$ as a function of can be obtained as a result of this stage.

$$J_K = \min_{\xi_K} ((a_K + b_K p_K + c_k p_k^2) + J_{K-1}(_{k-1}))$$

$$\xi_K = \xi_{K-1} + S_K \qquad K = 2,3,4,\ldots,6$$

(4.5.4)

At this stage, for a given ξ_K, the feasible values of Sk must be searched for to obtain ξ_{K-1} (4.5.4). Because J_{K-1} is a fjnction of ξ_{K-1}, the suboptimal solution up to the current state has been embedded in the last stage.

$$J_7 = \min_{\underline{0}} \left\{(829406.7 - 1184570p_7 + 196533.2p_7^2) + J_6(\xi_6)\right\}$$

$$\xi_7 = 0 = \xi_6 + S_7$$

(4.5.5)

At the last stage, $\xi_7 = 0$ because of the formulation of the a feasible S_7. Thus, the suboptimal policy can be obtained by embedding results up to stage 6 just as in previous stages. J_7 in the present is the minimum value of the objective function, while the optimal control sequence p has to be traced backward according to suboptimal values of the objective function at different stages.

In order to use the numerical method of SDP, a set of initial values for the control variables p must be given to start the searching process. In this work, all initial values of p start at their lower bounds, 0.35, and end with their common upper bounds, 0.90. Another numerical aspect which needs to be determined is the quantized increment between the bounds, pi and ti. Various values of increment have been used in this work. However, here only the Streeter-Phelps model is used because of its simplicity. The computational results are summarized in the section follows. (computer program listing is shown in Appendix)

4.6 Numerical results

Using the data given in Table 4.2.1, this water treatment problem is solved by the SDP method. The result are compared with Lee's [Lee, 1972], in which the gradient projection method ~~as~~ was used to solve the same problem. This is summarized in the following table: (Table 4.6.1)

Table 4.6.1 Numerical results of SDP with comparison to gradient projection method (%)

Reach	Gradient Projection Method	SDP 10 increments	SDP 50 increments	SDP 100 increments
1	65	62.5	62.5	69
2	54	50	50	50
3	42	45	44	41
4	90	91	90	90
5	--	--	--	--
6	90	92	92	94
7	50	50	50	50
Total cost	3,038,937	3,121,855	3,112,259	3,108,614

Approximately 0.5 minute, 2 minutes, and 5 minutes are needed for 10 increaments, 50 increaments and 100 increaments, respectively. In general, the more the increment steps, the better the solution is. However, more computation time is needed for the better solution.

4.7 Discussion

The result of the SDP technique on the water treatment via spatial dynamic programming is much better than the linear programming method, and is at about the same precision order as the gradient projection method. Besides, when using the gradient projection method, it is almost impossible to know how 'optimal' has been obtained because of the complex internal computation structure. This presents the problem of knowing when to terminate the optimizing process. On the other hand, the optimal is well understood throughout the optimizing process of the problem if SDP technique is applied. This is the promising feature SDP adopts from the straight-forward dynamic programming. As discussed in previous chapters, SDP can be applied on complex interacting systems such as computer network systems, digital communication systems, and several others differing from the water treatment problem solved here. This creates a huge dimension for SDP in the future.

CHAPTER 5
CONCLUSION

In this report, past research on SDP has been summarized. Also, a 7-reaches river pollution control problem has been solved by using SDP. The applications of SDP have been addressed, and some unanswered questions presented.

Optimal control over the large-scale system has been seen in the past twenty years and will become dominant in the future. Evolving under highly technological circumstances the modern control system becomes more complex. SDP seems to be a promising technique to handle these interacting systems, because of its uniqueness in global optimality. The only difficulty remaining is the formidable computational tasks invlved in dealing with the highly interacting dynamical systems. Some computational-oriented techniques, such as successive approximation, or grid-coarsed estimation have been implemented to overcome this difficulty. However, the use of these techniques will reach suboptimality rather than global optimality. Devising an algorithm by combining the simplicity of a computational task with the global optimality of SDP is a topic worth studying in the future.

On the other hand, decomposition principles are not 'concrete', as is found from past research. Proposing the most effective decomposition rule for SDP presents another challenge in the large-scale system optimization field.

After these drawbacks have been removed, the stochastic problems for the large-scale interacting system will be worth studying, thus opening up yet another challenging world for exploration.

REFERENCES

[A-1]: Araki, Mituhiko, 'Stability of Large-Scale Nonlinear Systems - Quadratic - Order Theory of Composite - System Method Using M-Matrices', IEEE Trans. on Automatic Control , Vol. AC-23, No. 2, 1978, pp. 129-148.

[A-2]: Arrow, K. J. and L. Hurwicz, 'Decentralization and Computation in Resource Allocation', in Essays in Economics and Econometrics , R. W. Dfonts, Ed.: Univ. North Carolina Press, 1960, pp. 34-104.

[A-3]: Aris, R.: Discrete Dynamic Programming, Blaisdell Publishing Company, New York, 1964.

[A-4]: Anton, J. J., B. Q. Friedlander, S. H. Javid, R. E. Larson, P. L. McEntire and T. L. Steding, 'Decentralized Control', Midterm Progress Report Prepared for Ballistic Missile Defense Systems Command, Systems Control, Inc., November 1978.

[B-1]: Bellman, R. E. and S. E. Dreyfus: Applied Dynamic ProgrammingPrinceton University Press, New Jersy (1962).

[C-1]: Cohen, Guy, 'Optimization by Decomposition and Coordination: a Unified Approach', IEEE Trans. Automatic Control, Vol. AC-23, No. 2, 1978, pp. 222-232.

[C-2]: Cline, T. B., and R. E. Larson, 'Decision and Control in Large-Scale Systems via Spatial Dynamic Programming', Proc. of Lawrence Symposium on Systems and Decision Sciences, Berkeley, California, October 1977.

[C-3]: Chong, C. Y., P. L. McEntire, and R. E. Larson, 'Decomposition of Mathematical Programming by Dynamic Programming', Proc. of Second Lawrence Symposium on Systems and Decision Sciences, Berkely, California, October 1978, pp. 215-221.

[C-4]: Camp, T. R., Water and Its Imputeries, Rainhold Publishing Co., New York, 1963

[D-1]: Dantzig, George B., Linear Programming and Extensions, Princeton University Press, New Jersy, 1963, Reprinted 1979, pp. 448-470.

[D-2]: Douglas, P. Looze, and Nils R. Sandell, JR, 'Hierarchical Control of Weakly-Coupled Systems', Automatica, Vol. 18, No. 4, pp. 467-471, 1982.

[H-1]: Howard, R. A., Dynamic Probabilistic SystemsNew York, Wiley 1971.

[J-1]: James A., Mathematical Models in Water Polution Control, New York, 1978

[K-1]: Kwatny, H. J., J. K. Sparse, F. M. Massimo, and A. D. Bhatt, 'Perturbation Methods in the Construction of Model Decomposition for Large Scale Systems Analysis', Drexel Univ., Rep., Aug. 1977.

[K-2]: Kokotivic', P. V., O'malley, R. E., and Sannuti, P., 'Singular Perturbations and Order Reduction in Control Theory - an Overview', Automatica, Vol. 12, pp. 123-132, March, 1976.

[K-3]: Kokotivic', P. V., and Singh, C., 'Optimization of Coupled Nonlinear Systems', Int. J. Contr., vol. 14, No. 1, pp. 51-64, 1971.

[K-4]: Korsak, A. J., and R. E. Larson, 'A Dynamic Programming Successive Approximations Techniquewith Convergence Proofs', Automatica , Vol. 6, pp. 253-260.

[K-5]: Kendric, D. A., Rao, H. S., and Wells, C. H., 'Optimal Operation of a System of Waste Treatment Facilities', Proc. IEEE Symposium on Adaptive Processes, Austin, Texas, 1970.

[L-1]: Li, Guangquan and Gordon K. F. Lee, 'Decentralized Control of Discrete-Time Large-Scale Systems by Dynamic Programming', Proc. 21st IEEE Conference on Decision and Control, Vol. 2, pp. 881-885, Dec. 1982, Orlando, Florida.

[L-2]: Lasdon, L. S. and J. D. Schoeffer, 'A Multilevel Technique for Optimization', in Proc. JACC, Troy, New York, 1965.

[L-3]: Larson, R. E. and A. J. Korsad, 'A Dynamic Programming Successive Approximations Technique with Convergence Proofs - Part I', Automatica, Vol. 6, pp. 245-252, March, 1970.

[L-4]: Larson Robert E, Paul L. McEntire, and Thomas L. Steding, 'Fundations of Spatial Dynamic Programming', Proc. 1st IEEE Intl. Conference Distributed Computing Systems, pp. 572-578, Oct. 1979, Huntsville, Alabama.

[L-5]: Lee, E. S. and E. H. Gray, 'Optimizing Complex Chemical Plants by Mathematical Modeling Techniques', Chemical Engineering, August 1967.

[L-6]: Li, Guangquan and Gordon K. F. Lee, 'Decentralized Control of Large Scale Systems with Dynamic Interconnected Subsystems', Int. J. Control, 1983, Vol. 37, No. 4, 775-786.

[L-7]: Larson, R. E., Luenberger, D. G., and Stengel, D. N., 'Optimization of Large-Scale Deterministic Systems Using Descriptor Variable Theory and Spatial Dynamic Programming', SCI Report, TM5168-9, Systems Control, Inc., 1978.

[L-8]: Lee, E. S. and P. D. Dand, 'Optimization of Water Quality by the Gradient Projection Method', Special Report Number 106, June, 1972, Kansas State University.

[L-9]: Loucks, D. P., C. S. Revelle and W. R. Lynn, 'Linear programming models for water pollution control', Management Science, 14, B-166, 1967

[M-1]: Mesarovic, M. D., Macko, D., and Takahara, Y., Theory of Hierarchical Multilevel Systems, N. Y., Academic Press, 1970.

[M-2]: Mahmound, M. S., 'Multilevel Systems Control and Applications: A Survey', IEEE Trans. Systems, Man, and Cybern., Vol. SMC-7, No. 3, pp. 125-143, March, 1977.

[M-3]: McEntire, P. L., C. Y. Chong, and R. E. Larson, 'A

Global Optimality Theorem for Spatial Dynamic Programming', Proc. of Second Lawrence Symposium on Systems and Decision Sciences, Berkeley, California, October 1978, pp. 341-345.

[O-1]: Ozguner, Umit, 'The Decentralized Servocompensator for Two Time-Scale Systems', Proc. 21st IEEE Conference on Decision and Control, Vol. 1, pp. 506-509, Dec. 1982, Orlando, Florida.

[P-1]: Palmer, James D. and Richard Saeks, The World of Large Scale Systems, IEEE Press, 1982, VII.

[S-1]: Sandell, N. R., Jr, P. Varaiya, M. Athans, and M. G. Safonov, 'Survey of Decentralized Control Methods for Large Scale Systems', IEEE Trans. Automatic Control, Vol. AC-23, No. 2, 1978, pp. 108-128.

[S-2]: Singh, M. G., Dynamical Hierarchical Control, North-Holland, 1980, pp. 23-26, pp. VII.

[S-3]: Singh, G. Madan, Jean-Pierre Elloy, R. Mezencer, and Neil Munro, Applied Industrial Control, Pergamon Press, 1980, pp. 423-427.

[S-4]: Stengel, D. N., D. G. Luenberger, R. E. Larson, and T. B. Cline, 'A Descriptor Variable Approach to Modeling and Optimization of Large-Scale Systems', Final Report to the Department of Energy, Systems Control, Inc. February, 1979.

[T-1]: Takahara, Y., 'Multilevel Approach to Dynamic Optimization', Syst. Res. Center. Case Western Reserve University, Cleveland, OH, Rep.

Reserve University, Cleveland, OH, Rep. SRC-50-C-64-18, 1964.

[T-2]: Tamura, H., 'Decomposition Techniques in Large Scale Systems with Applications', _Systems and Control_, Vol. 17, 6, 1973 (in Japanese).

[T-3]: Tamura, H., 'Decentralized Optimization for Distributed-Big Models of Discrete Systems', _Automatica_, Vol. 11, 6, 1975, pp. 593-602.

[W-1]: Wismer, D. A. (ed.) _Optimization Methods for Large Scale Systems_, N. Y., McGraw-Hill, 1971.

APPENDIX

```
FILE:  DYNAMIC1 WBASIC   A1                 KANSAS STATE UNIVERSITY VM/SP CMS

00010 DIM PMAX(7),PMIN(7),BOMAX(7),BOMIN(7),BFMAX(7),BFMIN(7)
00020 DIM YMAX(7),YMIN(7),ZMAX(7),ZMIN(7)
00030 DIM T(7),QW(7),QT(7),Q(7),BW(7),BT(7),CW(7),CT(7),AS(7),D(7)
00040 DIM KK(7,3),CC(7,3)
00050 DIM Z(2501),ZL(2501),Y(2501),P(7,2501),COST(7,2501),LSTATE(7,2501)
00060 BIG=99999999
01000 FOR I=1 TO 7
01010 PMAX(I)=0.95
01020 PMIN(I)=0.30
01022 IF I=5 THEN PMAX(I)=0
01024 IF I=5 THEN PMIN(I)=0
01030 READ T(I),QW(I),QT(I),Q(I),AS(I),D(I),CW(I),CT(I),BT(I)
01040 READ KK(I,1),KK(I,2),KK(I,3),BW(I),CC(I,1),CC(I,2),CC(1,3)
01050 NEXT I
02000 I=1
02002 LOOP
02004 GOSUB 15500
02005 GOSUB 19000
02006 ON I GOTO 2020,2020,2160,2020,2160,2160,2160
02020 P=PMAX(I)
02030 STEPP=0.001
02040               LOOP
02050     BO=FN_BO_P
02060     BF=FN_BF_BO
02062 CO=0
02064 CL=0
02070      CO=FN_CC
02080      DF=FN_DF
02090      TC=FN_TC
02100      DMAX=FN_DMAX
02106                         IF TC<=0 THEN DMAX=AS-CC
02110                        IF TC>T THEN DMAX=DF
02120                        IF D>= DMAX THEN 2130 ELSE 2140
02130 PMIN(I)=P
02140 P=P-STEPP
02145                        IF P<.3 THEN QUIT
02150              ENDLOOP
02160  PRINT 'AT STAGE : ';I
02170 REM PRINT
02180  PRINT PMIN(I);'<=P<=';PMAX(I)
02181 REM PRINT
02190 ON I GOTO 2200,2200,2300,2200,2300,2400,2400
02200 P=PMIN(I)
02210 BOMIN(I)=FN_BO_P
02220 P=PMAX(I)
02230 BOMAX(I)=FN_BO_P
02240 GOTO 2500
02300 T1=Q(I-1)
02302 T2=Q(I-2)
02304 BOMAX(I)=(BFMAX(I-1)*T1+BFMAX(I-2)*T2+BW*(1-PMIN(I))*QW)/Q
02310 BOMIN(I)=(BFMIN(I-1)*T1+BFMIN(I-2)*T2+BW*(1-PMAX(I))*QW)/Q
02320 GOTO 2500
02400 T1=Q(I-1)
02404 BOMAX(I)=(BFMAX(I-1)*T1+BW*(1-PMAX(I))*QW)/Q
02410 BOMIN(I)=(BFMIN(I-1)*T1+BW*(1-PMIN(I))*QW)/Q
02420 GOTO 2500
02500 IF BOMAX(I)<BOMIN(I) THEN 2510 ELSE 2540
```

```
FILE:  DYNAMIC1 WBASIC   A1             KANSAS STATE UNIVERSITY VM/SP CMS

02510         TEM=BOMIN(I)
02520         BOMIN(I)=BOMAX(I)
02530         BOMAX(I)=TEM
02540  PRINT BOMIN(I);'<=BO=<';BOMAX(I)
02541 REM PRINT
02550 BO=BOMIN(I)
02560 BFMIN(I)=FN_BF_BO
02570 BO=BOMAX(I)
02580 BFMAX(I)=FN_BF_BO
02590 IF BFMAX(I)<BFMIN(I) THEN 2600 ELSE 2630
02600         TEM=BFMIN(I)
02610         BFMIN(I)=BFMAX(I)
02620         BFMAX(I)=TEM
02630  PRINT BFMIN(I);'<=BF<=';BFMAX(I)
02631 REM PRINT
02640 ON I GOTO 2650,2650,2680,2650,2680,2680,2680
02650 YMAX(I)=-BFMIN(I)*Q(I)
02660 YMIN(I)=-BFMAX(I)*Q(I)
02670 GOTO 2730
02680 YMIN(I)=-ZMAX(I-1)
02690 YMAX(I)=-ZMIN(I-1)
02700 GOTO 2730
02710 YMIN(I)=+BFMAX(I-1)*Q(I-1)
02720 YMAX(I)=+BFMIN(I-1)*Q(I-1)
02730 IF YMIN(I)>YMAX(I)THEN 2732 ELSE 2738
02732 TEM=YMIN(I)
02734 YMIN(I)=YMAX(I)
02736 YMAX(I)=TEM
02738 PRINT YMIN(I);'<=Y<=';YMAX(I)
02740 ON I GOTO 2750,2780,2810,2802,2810,2810,2840
02750 T1=YMIN(I)
02760 T2=YMAX(I)
02770 GOTO 2860
02780 T1=YMIN(I)+YMIN(I-1)
02790 T2=YMAX(I)+YMAX(I-1)
02792 REM PRINT 'YMIN(';I;')=';YMIN(I);'YMIN(';I-1;')=';YMIN(I-1);
02794 REM PRINT 'YMAX(';I;')=';YMAX(I);'YMAX(';I-1;')=';YMAX(I-1)
02800 GOTO 2860
02802 T1=YMIN(I)+ZMIN(I-1)
02804 T2=YMAX(I)+ZMAX(I-1)
02806 GOTO 2860
02810 T1=-BFMIN(I)*Q(I)
02820 T2=-BFMAX(I)*Q(I)
02830 GOTO 2860
02840 T1=0
02850 T2=0
02860 ZMIN(I)=T1
02870 ZMAX(I)=T2
02872 IF ZMAX(I)<ZMIN(I) THEN 2874 ELSE 2890
02874 TEM=ZMAX(I)
02876 ZMAX(I)=ZMIN(I)
02878 ZMIN(I)=TEM
02890  PRINT ZMIN(I);'<=Z<=';ZMAX(I)
02891 REM PRINT
02900 GOSUB 15700
02904 IF I=7 THEN QUIT
02906 I=I+1
```

```
FILE:  DYNAMIC1 WEASIC   A1            KANSAS STATE UNIVERSITY VM/SP CMS

02908 ENDLOOP
02910 GOTO 35000
15000 DEF FN_BO_P
15010      FN_BO_P=(BT*QT+BW*(1-P)*QW)/Q
15020 FNEND
15030 DEF FN_CO
15040      FN_CO=(CL*QL+CT*QT+CW*QW)/Q
15050 FNEND
15060 DEF FN_TC
15070              TEM1=1/(K2-K1)
15080              TEM2=K1*BO-(K2-K1)*(AS-CO)
15090              TEM3=K2/(K1**2*BO)
15092 IF TEM2*TEM3 < 0 THEN 15094 ELSE 15100
15094 FN_TC=BIG
15096 GOTO 15200
15100      FN_TC=TEM1*LOG(TEM2*TEM3)
15200 FNEND
15210 DEF FN_DMAX
15212 IF TC=BIG THEN 15214 ELSE 15220
15214 FN_DMAX=BIG
15216 GOTO 15230
15220      FN_DMAX=K1*BO/(K2*EXP(-K1*TC))
15230 FNEND
15240 DEF FN_BF_BO
15250      FN_BF_BO=BO*EXP(-K1*T)
15260 FNEND
15270 DEF FN_DF
15280              TEM1=K1/(K2-K1)
15290              TEM2=BO
15300              TEM3=EXP(-K1*T)-EXP(-K2*T)
15310              TEM4=(AS-CO)*EXP(-K2*T)
15320      FN_DF=TEM1*TEM2*TEM3+TEM4
15330 FNEND
15340 DEF FN_BC_BF
15350      FN_BC_BF=BF*EXP(K1*T)
15360 FNEND
15370 DEF FN_P_BO
15380      FN_P_BO=-(BO*Q-BT*QT-BW*QW)/(BW*QW)
15390 FNEND
15400 DEF FN_COST_P
15410      FN_COST_P=C1+C2*P+C3*P**2
15420 FNEND
15450 DEF FN_CF_1
15455                 TEM1=AS(I-1)*(1-EXP(-KK(I-1,2)*T(I-1)))
15460                 TEM2=KK(I-1,1)/(KK(I-1,2)-KK(I-1,1))*BO1
15465                 TEM3=EXP(-KK(I-1,1)*T(I-1))-EXP(-KK(I-1,2)*T(I-1))
15470                 TEM4=CO1*EXP(-KK(I-1,2)*T(I-1))
15475      FN_CF_1=TEM1-TEM2*TEM3+TEM4
15480 FNEND
15482 DEF FN_CF_2
15484                TEM1=AS(I-2)*(1-EXP(-KK(I-2,2)*T(I-2)))
15486                TEM2=KK(I-2,1)/(KK(I-2,2)-KK(I-2,1))*BO2
15488                TEM3=EXP(-KK(I-2,1)*T(I-2))-EXP(-KK(I-2,2)*T(I-2))
15490                TEM4=CO2*EXP(-KK(I-2,2)*T(I-2))
15492      FN_CF_2=TEM1-TEM2*TEM3+TEM4
15494 FNEND
15500 T=T(I)
```

```
FILE:  DYNAMIC1 WBASIC    A1              KANSAS STATE UNIVERSITY VM/SP CMS

15510 QW=QW(I)
15520 QT=QT(I)
15530 Q=Q(I)
15540 BW=BW(I)
15550 BT=BT(I)
15560 CW=CW(I)
15570 CT=CT(I)
15580 AS=AS(I)
15590 D=D(I)
15600 K1=KK(I,1)
15610 K2=KK(I,2)
15620 K3=KK(I,3)
15630 C1=CC(I,1)
15640 C2=CC(I,2)
15650 C3=CC(I,3)
15660 RETURN
15700 KZ=1
15710 Z=ZMIN(I)
15720 STEPZ=(ZMAX(I)-ZMIN(I))/30
15721 IF I=7 THEN STEPZ=BIG
15722  REM PRINT 'STEP Z=';STEPZ
15730 LOOP
15740 IF I=5 THEN PRINT 'Z=';Z
15750      KY=1
15760      Y=YMIN(I)
15770      STEPY=(YMAX(I)-YMIN(I))/30
15772 REM PRINT 'STEP Y=';STEPY
15780      COST(I,KZ)=BIG
15782 IF I=1 THEN Y=Z
15783 IF I=5 THEN Y=-Z/EXP(-K1*T)
15784 IF I=1 THEN STEPY=BIG
15785 IF I=5 THEN STEPY=BIG
15790                                   LOOP
15800 IF I=5 THEN PRINT 'Y=';Y
16000 IF I=1 THEN 16002 ELSE 16010
16002 LSTATE(I,KZ)=0
16004 COSTL=0
16006 GOTO 16040
16010 IF I=2 OR I=4 THEN ZL=Z-Y ELSE ZL=-Y
16020     GOSUB 25100
16030     IF TEST_ZL$='FALSE' THEN 16031 ELSE 16040
16031 IF I=5 THEN 16032 ELSE 16038
16032 PRINT 'ZMIN(';I-1;')=';ZMIN(I-1);'ZL=';ZL;
16034 PRINT 'ZMAX(';I-1;')=';ZMAX(I-1)
16036 PRINT '***** INADMISSIBLE *****'
16038 GOTO 18220
16040 IF I<3 OR I=4 THEN BF=-Y/Q ELSE BF=-Z/Q
16045 IF I=7 THEN PAUSE
16050     GOSUB 25200
16055 IF I=7 THEN PAUSE
16060     GOSUB 25000
16065   IF I=5 THEN GOSUB 19010
16068 GOTO 16080
16070     IF TEST_D$='FALSE' THEN 16071 ELSE 16080
16071 IF I=5 THEN 16072 ELSE 16076
16072 PRINT 'DMAX=';DMAX;'D=';D;,'DMAX>D'
16074 PRINT '***** INADMISSIBLE *****'
```

```
FILE:  DYNAMIC1 WBASIC   A1              KANSAS STATE UNIVERSITY VM/SP CMS

16076 GOTO 18220
16080     IF I=4 OR I<3 THEN P=FN_P_BO ELSE P=-(-Y+BO*Q-BW*CW)/(BW*QW)
16090     IF I=5 THEN P=0
16100     IF PMIN(I)>P OR P>PMAX(I) THEN 16101 ELSE 16110
16101 IF I=5 THEN 16102 ELSE 16108
16102 PRINT 'PMIN(';I;')=';PMIN(I);
16104 PRINT 'P=';P;'PMAX(';I;')=';PMAX(I)
16106 PRINT '***** INADMISSIBLE *****'
16108 GOTO 18220
16110   COST=FN_COST_P
18175 COSTY=COST+COSTL
18180          IF COSTY<COST(I,KZ) THEN 18200 ELSE 18212
18200        COST(I,KZ)=COSTY
18205        P(I,KZ)=P
18210      LSTATE(I,KZ)=KZL
18211 IF I=5 THEN 18212 ELSE 18220
18212  PRINT 'Y=';Y;'P=';P;'COST=';COST;'COSTY=';COSTY;'COSTL=';
18213  PRINT COST(I-1,LSTATE(I,KZ)),'KZ=';KZ,'LSTATE=';KZL,'Z=';Z,'ZL=';ZL
18214REM PRINT
18220 Y=Y+STEPY
18230             IF Y>=YMAX(I) THEN QUIT
18240 KY=KY+1
18250                                   ENDLOOP
18252REM PRINT 'CURRENT STATE : Z(';KZ;')=';Z(KZ);
18253REM PRINT
18254REM PRINT 'COST = ';COST(I,KZ)
18256 REM PRINT 'LAST STATE ';LSTATE(I,KZ);'COST FROM PREVIOUS STAGES';
18257 REM PRINT (COST(I-1,LSTATE(I,KZ))+COST(I-1,LSTATE(I,KZ)+1))/2
18258 REM PRINT '--------------------------------------------------'
18259 Z(KZ)=Z
18260 Z=Z+STEPZ
18270              IF Z>ZMAX(I) THEN QUIT
18280 KZ=KZ+1
18290 ENDLOOP
18292 IF I=5 THEN 35000
18300 IF I<7 THEN 18302 ELSE 18352
18302 NUMSTA=KZ
18305 ZL(NUMSTA+1)=Z(NUMSTA)
18310 LOOP
18320      ZL(NUMSTA)=Z(NUMSTA)
18330      NUMSTA=NUMSTA-1
18340      IF NUMSTA=0 THEN QUIT
18350 ENDLOOP
18352 GOSUB 19050
18360 RETURN
19000 REM PRINT 'REACH=';I;'T=';T(I);'QW=';QW(I);'QT=';QT(I);'Q=';Q(I);
19002 REM PRINT 'AS=';AS(I);'D=';D(I);'CW=';CW(I)
19004 REM PRINT '      CT=';CT(I);'BT=';BT(I);'K1=';KK(I,1);'K2=';
19005 REM PRINT KK(I,2)
19006 REM PRINT 'K3=';KK(I,3);'BW=';BW(I);'C1=';CC(I,1);'C2=';CC(I,2);
19007 REM PRINT 'C3=';CC(I,3)
19008 REM PRINT
19009 RETURN
19010 PRINT 'P=';P,'BO=';BO,'BF=';BF,'CO=';CO,'DO=';AS-CO,'DF=';DF,
19012 PRINT 'TC=';TC,'T=';T,'D=';D,'DMAX=';DMAX
19013 PRINT
19014 RETURN
```

```
FILE:  DYNAMIC1 WBASIC   A1              KANSAS STATE UNIVERSITY VM/SP CMS

19050 PRINT
19052  PRINT '************************************************
19054  PRINT '*          AT     STAGE';I;'                    *'
19055 PRINT '************************************************'
19056 FOR K=1 TO KZ
19057 PRINT 'Z(';K;')=';Z(K);
19058 IF COST(I,K)>999 THEN 19059 ELSE 19061
19059 PRINT 'INADMISSIBLE****************************************'
19060 GOTO 19068
19061 COST1=COST(I-1,LSTATE(I,K))
19062 PRINT 'COST(';I;',';K;')=';COST(I,K);
19064 PRINT'LSTATE=';LSTATE(I,K);'COSTL=';COST1;'PL=';P(I-1,LSTATE(I,K))
19066 PRINT 'CURRENT COST=';COST(I,K)-COST1;'CURRENT P=';P(I,K)
19068 NEXT K
19070 RETURN
20000 DATA .235,5,1355,1360,10.2,3.2,1,9.5,1.66,.31,1.02,.02,248
20010 DATA .6677246,-2.128173C,1.687988
20020 DATA 1.33,37,1290,1327,9.95,2.45,1,8,.68,.41,.6,.03,408
20030 DATA 1.086456,-2.518554,2.862304
20040 DATA 1.087,8,1360,2695,9,2,1,0,0,.36,.63,.04,240
20050 DATA .2358398,-.457077,.6746979
20060 DATA 2.067,14,296,310,9.54,3.54,1,9.7,1,.35,.09,.04,1440
20070 DATA .4561768,-.8029786,1.1875
20080 DATA .306,0,310,3005,9,2.5,0,0,0,.34,.72,.05,0
20090 DATA 0,0,0
20100 DATA 1.05,26,0,3031,8.35,2.35,1,0,0,.35,.14,.06,2180
20110 DATA .6172485,-1.280029,1.759033
20120 DATA 6.13,41,0,3072,8.17,4.17,1,0,0,.3,.02,0,279
20130 DATA .8294067,-1.18457,1.965332
25000 DF=FN_DF
25010 TC=FN_TC
25020 DMAX=FN_DMAX
25030                            IF TC<=0 THEN DMAX=AS-CO
25040                            IF TC>=T THEN DMAX=DF
25050 IF D >= DMAX THEN TEST_D$='TRUE' ELSE TEST_D$='FALSE'
25060 RETURN
25100 IF ZMIN(I-1)<=ZL AND ZL<=ZMAX(I-1) THEN 25130 ELSE 25110
25110 TEST_ZL$='FALSE'
25120 GOTO 25195
25130 TEST_ZL$='TRUE'
25140 KZL=1
25150       LOOP
25160       IF ZL(KZL)<= ZL AND ZL<=ZL(KZL+1) THEN QUIT
25170       KZL=KZL+1
25180       ENDLOOP
25192 COSTL=COST(I-1,KZL)
25194 IF KZL=31 THEN COSTL=BIG
25195 RETURN
25200 BO=FN_BO_BF
25202 ON I GOTO 25210,25210,25220,25210,25220,25230,25230
25210 CL=0
25212 QL=0
25214 CO=FN_CO
25216 GOTO 25290
25220 GOSUB 26000
25222 CO=FN_CO
25224 GOTO 25290
```

```
FILE:  DYNAMIC1 WBASIC    A1                KANSAS STATE UNIVERSITY VM/SP CMS

25230 CL=0
25232 CT=0
25234 GOSUB 26050
25236 CO=FN_CO
25238 GOTO 25290
25240 CL=0
25290 RETURN
26000 P2=P(I-2,LSTATE(I-1,KZL))
26002 B02=(BT(I-2)*QT(I-2)+BW(I-2)*QW(I-2)*(1-P2))/Q(I-2)
26004 C02=(CT(I-2)*QT(I-2)+CW(I-2)*QW(I-2))/Q(I-2)
26006 CF_2=FN_CF_2
26008 CL=CF_2
26010 QL=Q(I-2)
26050 P1=P(I-1,KZL)
26052 B01=(BT(I-1)*QT(I-1)+BW(I-1)*QW(I-1)*(1-P1))/Q(I-1)
26054 CO1=(CT(I-1)*QT(I-1)+CW(I-1)*QW(I-1))/Q(I-1)
26056 CF_1=FN_CF_1
26058 CT=CF_1
26060 QT=Q(I-1)
26061 IF I=5 THEN GOSUB 27000
26062 RETURN
27000 PRINT 'STATE1=';KZL;'STATE2=';LSTATE(I-1,KZL)
27010 PRINT 'P1=';P1;'P2=';P2
27020 PRINT 'B01=';B01;'B02=';B02;'C01=';C01;'C02=';C02
27030 PRINT 'CF1=';CF_1;'CF2=';CF_2
27040       FOR J=I-2 TO I-1
27050       PRINT 'I=';J;'BT(';J;')=';BT(J);'CT(';J;')=';CT(J);
27060       PRINT 'BW(';J;')=';BW(J);'CW(';J;')=';CW(J);
27070       PRINT 'QT(';J;')=';QT(J);'QL(';J;')=';QL(J);
27080       PRINT 'Q(';J;')=';Q(J)
27090       NEXT J
27100 RETURN
35000 END
```

SPATIAL DYNAMIC PROGRAMMING ON THE WATER TREATMENT PROBLEM

by

JONG-WOEI WHANG

B.S. (Industrial Engineering)

National Tsing Hua Univerisity

Hsin Chu, Taiwan, Republic of China, 1978

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the requirement for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

Kansas State University

Manhattan, Kansas

1983

ABSTRACT

Computational difficulty increases exponentially as the dimension of problem grows. This predicament prevails in many fields which deal with the large-scale problems, such as traffic network, digital communication, ecological systems, and economic systems. Linear programming, dynamic programming, and other mathematical techniques have been used to solve this sort of problems, but fail to achieve progressive results. Because these methods, either fail to obtain global solution, or can not be applied to non-linear or non-serial structure problems, they would be too difficult to solve the large-scale problems of the real world.

In this research, a new approach of dynamic programming, called Spatial Dynamic Programming (SDP) is employed to overcome the difficulties. A seven-reached water treatment problem is solved with the SDP technique. Although this problem can be sloved by linear programming or gradient projection method, neither of them can obtain solution for a straightforward problem. On the other hand, SDP can solve the same problem without simplification. This result not only proclaims SDP's competitiveness to other methods, but also reveals SDP's uniqueness in dealing with a highly interacted problem.

Computational work of a large-scale system, although greatly reduced by using the SDP technique, is still a

formidable task because of the interacting nuture of the system. Studies need to be done to find algorithms to carry out SDP technique more efficiently. Thus, creating a challenging aspect of the continuation of this research.