PARALLEL ALGORITHMS FOR A CLASS OF CONVEX OPTIMIZATION PROBLEMS

by

Rong-Jaye Chen

Computer Sciences Technical Report #731

December 1987

PARALLEL ALGORITHMS FOR A CLASS OF CONVEX OPTIMIZATION PROBLEMS

by

Rong-Jaye Chen

A thesis submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN-MADISON

1987

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to my thesis advisor Professor Robert R. Meyer for his guidance, advice and encouragement in the preparation of this thesis.

I would also like to thank Professor Olvi L. Mangasarian and Professor Stephen M. Robinson for reading drafts of the thesis, and Professor John Stirkwerda and Professor Kam Tsui for being members of the Examination Committee.

I dedicate this thesis to my parents, both of whom passed away during my study years abroad, and to my wife Hai-Tseu, who had been proofreading the drafts all along. Without their love and support, this work would not have been accomplished.

This research was supported by the National Science Foundation under grant DCR-8503148 and by the Air Force under grant AFOSR-86-0194.

PARALLEL ALGORITHMS FOR A CLASS OF CONVEX OPTIMIZATION PROBLEMS

Rong-Jaye Chen

Under the supervision of Professor Robert R. Meyer

ABSTRACT

This thesis is principally concerned with a piecewise-linear trust region method for solving a class of structured convex optimization problems, which includes the traffic assignment problems. Piecewise-linear approximation of nonlinear convex objective functions in linearly constrained optimization produces subproblems that may be solved as linear programs. This approach to approximation may be used for nonseparable as well as separable functions, and for the former class (the focus of this thesis), it lies between linear and quadratic approximation with regard to its accuracy. In order to have additional control of the accuracy of the piecewise-linear approximation, we consider two devices : rectangular trust regions and dynamic scaling. The use of rectangular trust regions in conjunction with the type of piecewise-linear approximation considered here actually serves to simplify rather than complicate the approximating problems. This is a result of the equivalence of the trust region and the use of a limited number of segments of comparable size in the approximation. The approach to dynamic scaling considered here may be applied to problems in which each objective function term is a convex function of a linear function of the variables. This scaling device allows the algorithm to adjust the approximation between an underestimating function (corresponding to a linear approximation) and an overestimating function (the non-separable analog of the overestimate associated with separable approximation of convex functions.) The scaling factor is adjusted in accordance with the acceptance criteria associated with the trust region method.

Another emphasis of this thesis is the development of parallel algorithms suited to distributed computing and the comparison of the relative efficiencies of these algorithms on different architectures. Computational experience is cited for some large-scale problems arising from traffic assignment applications. The algorithms considered here also have the property that they allow such problems to be decomposed into a set of smaller optimization problems at each major iteration. These smaller problems correspond to linear single-commodity networks, and may be solved in parallel. Results are given for the distributed solution of these problems on the CRYSTAL multicomputer.

TABLE OF CONTENTS

	ACKNOWLEDGEMENTS	ii
	ABSTRACT	iii
Chapter 1	INTRODUCTION	
1.1	Piecewise-Linear Approximations	1
1.2	Parallel Algorithms	3
1.3	Nonlinear Multicommodity Problems (MCP)	5
1.4	Traffic Assignment Problems (TAP)	7
Chapter 2	REVIEW OF PREVIOUS WORK	
2.1	The Frank-Wolfe Method (FW)	12
2.2	The Reduced Gradient Method (RG)	12
2.3	The Piecewise-Linear Method (PL)	15
Chapter 3	SCALED SEPARABLE APPROXIMATIONS	
3.1	Introduction	17
3.2	Examples of Scaled Separable Approximations	18
3.3	Properties of Scaled Separable Functions	22
3.4	Extension to General Linear Case	26
Chapter 4	TRUST REGION METHODS	
4.1	Introduction	29
4.2	A Linear Trust Region Algorithm (LTR)	30
4.3	A Piecewise-Linear Trust Region Algorithm (PLTR)	36
Chapter 5	A SCALED PIECEWISE-LINEAR TRUST REGION ALGORITHM	
5.1	A Sketch of the SPLT Algorithm	43
5.2	Notions and Notations	49
5.3	The SPLT Algorithm	51

5.4	A Convergence Theorem for SPLT	53
Chapter 6	SEQUENTIAL METHODS	
6.1	A Block Gauss-Seidel Algorithm (BGS)	56
6.2	A Convergence Theorem for BGS	59
6.3	A Comparison of Sequential Algorithms	62
Chapter 7	PARALLEL METHODS	
7.1	The Crystal Multicomputer	65
7.2	A Parallel Block Jacobi Algorithm (PBJ)	67
7.3	A Parallel Block Gauss-Seidel Algorithm (PBGS)	74
7.4	Performance for PBGS	79
Chapter 8	DIRECTIONS FOR FURTHER RESEARCH	
8.1	Communication Routing Problems	89
8.2	Parallel Algorithms for Multiprocessors	91
	BIBLIOGRAPHY	94

CHAPTER 1

INTRODUCTION

This thesis is primarily concerned with methods of solution for a class of convex optimization problems. In Chapter 1, we introduce a class of structured problems. A review of previous work is presented in Chapter 2. Chapter 3 brings in the theoretical grounds of scaled separable approximations, and Chapter 4 illustrates trust region methods. Accordingly, Chapter 5 incorporates the previous two chapters and elaborates on a scaled piecewise-linear trust region algorithm. We discuss sequential methods in Chapter 6 and discuss parallel methods in Chapter 7. We conclude the thesis by directions for further research in Chapter 8.

1.1 PIECEWISE-LINEAR APPROXIMATIONS

Piecewise-linear approximation of convex nonlinear objective functions in linearly constrained optimization has the nice property of producing subproblems that may be solved as linear programs. This approach to approximation may be used for nonseparable as well as separable functions, and this thesis deals with nonseparable convex objectives that are sums of terms of the form $f_j(\mathbf{c}_j \cdot \mathbf{x})$, where f_j is a continuously differentiable convex function defined on R^1 and $\mathbf{c}_j \cdot \mathbf{x}$ is a linear function of the problem variables $\mathbf{x}$. Although such a problem could be transformed into a separable problem by substitutions of the form $t_j = \mathbf{c}_j \cdot \mathbf{x}$ and the concatenation of this equation to the constraints, this transformation can have undesirable effects. For the multicommodity problems considered below, for example, it would introduce additional coupling between variables and thereby destroy the block structure of the constraints, and this new constraint would also have a different character from the other constraints, resulting in the destruction of the network nature of the initial constraints. Thus, we will demonstrate how objective functions of this type may be treated directly without the introduction of new constraints.

Piecewise-linear approximation lies between linear and quadratic approximation with regard to its accuracy. In order to control the error in the piecewise-linear approximation , we introduce two devices : rectangular trust regions and dynamic scaling. The use of rectangular trust regions in conjunction with this type of piecewise-linear approximation was first described in [Meyer 85]. With this approach, trust region constraints serve to simplify rather than to complicate the approximating subproblem, because the equivalence between the trust region constraints and the use of a limited number of piecewise-linear segments in the approximation allows the former to be handled implicitly. The approach to dynamic scaling considered here is a new approximation tool that takes advantage of the assumed form of the objective function terms. This device allows the algorithm to adjust the approximation between an underestimating function (corresponding to a linear approximation) and an overestimating function (the nonseparable analog of the overestimating property inherent in separable approximation of convex functions.) The scaling factor is adjusted in accordance with the acceptance criteria associated with the trust region method.

For notational simplicity we develop the theory below in the context of objective functions that are sums of terms of the form $f_j(x_{1j}+\cdots+x_{kj})$, i.e., each term involves the sum of a fixed number (namely, k) of variables. The extension of these results to the case in which the argument is a linear function of $\mathbf{x}$ is straightforward. We also consider a specific block structure of the constraints in order to emphasize the decomposition that is possible when piecewise-linear approximation is used appropriately in that context. However, the theory as presented is also independent of the nature of the linear constraints.

Computational experience is cited for some large-scale problems arising from traffic assignment applications(see section 1.4). The algorithm considered in this thesis has the property that it allows such problems to be decomposed into smaller optimization problems at each major iteration. These smaller problems may be solved in parallel, and computational results are given for the distributed solution of these problems on the CRYSTAL multicomputer.

1.2 PARALLEL ALGORITHMS

Most large-scale optimization problems arising from real-world applications can be decomposed into quasi-independent subproblems (corresponding to time periods, geographic districts, physical or logical commodities, etc.), allowing the possibility of attack via iterative algorithms that exhibit a high degree of parallelism. Theoretical research into decomposition methods for large-scale optimization dates back to [Dantzig and Wolfe 60], but the absence of computer hardware capable of exploiting the parallelism inherent in these methods has long discouraged potential research in this area. With the advent of multicomputers and multiprocessors, research into new decomposition methods is not merely stimulated by the new algorithmic possibilities motivated by these distributed environments, but is further dictated by the need to achieve the speedups made possible by these architectures.

Optimization problems related to networks lend themselves particularly well to this type of research, not only because they are large-scale and arise in a multitude of diverse applications, but also because they generally can be partitioned into network subproblems that may be solved by very fast techniques. In this thesis, very promising results have been obtained through the use of CRYSTAL on nonlinear multicommodity problems(see next section). The methods in essence replace the original optimization problem at each major iteration by an approximation consisting of a set of linked subproblems that may be solved in parallel by temporarily ignoring the coupling between subproblems (which may occur in the objective function and/or the constraints). After this parallel phase, there is a coordination phase in which results from the subproblems are combined, and the linkages between subproblems are taken into account(see Fig. 1.1).

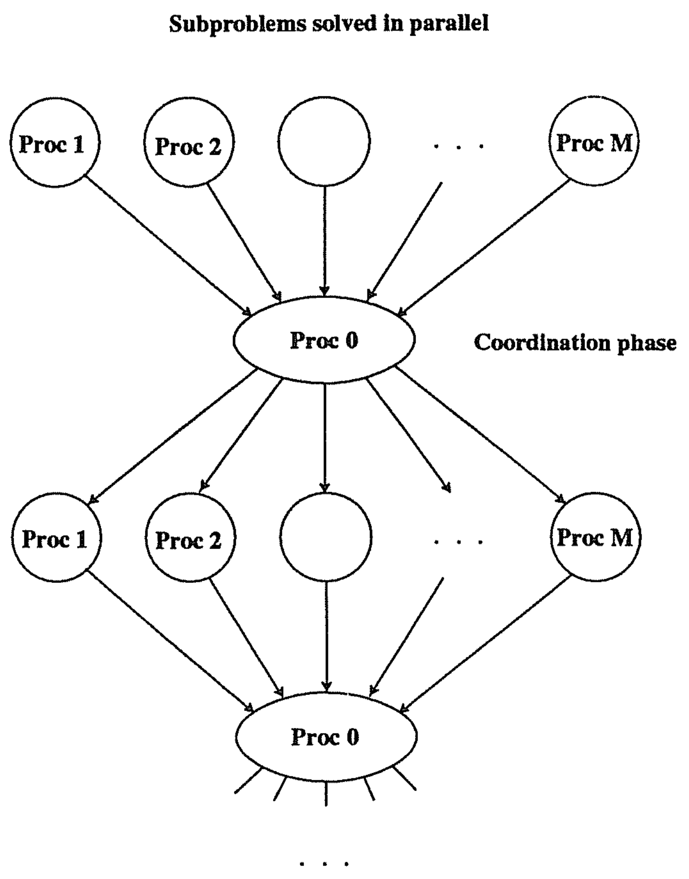


Figure 1.1 A Basic Parallel Algorithm

A significant challenge at this stage is to keep the processors busy doing useful work while the coordinating phase is in progress. The major thrusts of our research have thus been the development of procedures for (1) splitting large-scale problems into quasi-independent subproblems and (2) performing the coordinating phase so as to maximize overall parallel efficiency.

1.3 NONLINEAR MULTICOMMODITY PROBLEMS

In this section we will develop a model for a class of problems that may be attacked by the algorithms to be described below. An example of this problem class is the nonlinear multicommodity problem. Consider a directed network E of m nodes and n arcs. Let A be the node-arc matrix of E and k be the number of commodities sharing the network. Let x_{qj} denote the value of flow on arc j corresponding to commodity q. Let $\mathbf{x}_q$ and $\mathbf{b}_q$ denote the flow vector and supply-demand vector respectively of commodity q. If f_j $(j=1,...,n)$ is a set of continuously differentiable convex functions corresponding to the n arcs of the network, the corresponding multicommodity problem may be written as

$$\begin{array}{llr}
\min \sum_{j=1}^{n} f_j(x_{1j}+x_{2j}+\cdots+x_{kj}) & & \\
s.t. \quad \mathrm{A}\mathbf{x}_1 & = \mathbf{b}_1 & \\
\qquad\quad \mathrm{A}\mathbf{x}_2 & = \mathbf{b}_2 & \\
\qquad\qquad \cdot & \cdot & \\
\qquad\qquad\quad \cdot & \cdot & \text{(MCP)} \\
\qquad\qquad\qquad \cdot & \cdot & \\
\qquad\qquad\qquad\quad \mathrm{A}\mathbf{x}_k & = \mathbf{b}_k & \\
x_{qj} \geq 0, \; q=1,2,...,k, \; j=1,2,...,n & &
\end{array}$$

For notational convenience, we define

$\mathbf{x}_q := (x_{q1}, x_{q2}, \cdots, x_{qn}) \, \varepsilon \, R^n$ (flow vector or block of variables of commodity

q);

$\mathbf{x}_{.j} := (x_{1j}, x_{2j}, \cdots , x_{kj}) \varepsilon R^k$ (flow vector in arc j);

$\mathbf{x} := (\mathbf{x}_1, \mathbf{x}_2, \cdots , \mathbf{x}_k) \varepsilon R^{kn}$ (full set of flow vectors);

$f(\mathbf{x}) := \sum_{j=1}^{n} f_j (\sum_{q=1}^{k} x_{qj})$ (cost function);

$\Omega_q := \{ \mathbf{x}_q \varepsilon R_+^n \mid A\mathbf{x}_q = \mathbf{b}_q \}$ (feasible flow region of commodity q);

$\Omega := \{ \mathbf{x} \varepsilon R_+^{kn} \mid A\mathbf{x}_q = \mathbf{b}_q, \text{for } q=1,2,...,k \}$ (feasible flow region).

While nonlinear multicommodity problems furnish a good example of problems of the class MCP, the algorithms to be derived are in fact applicable to more general problems, for example the matrices A may differ for each $\mathbf{x}_j$ and the objective may be of the form $\sum_{j=1}^{J} f_j (\mathbf{c}_j \cdot \mathbf{x})$.

Solution Algorithms

Network flow problems of the form (MCP) include computer network design [Cantor and Gerla 74], [Magnanti and Wong 84], traffic assignment [Bertsekas and Gafni 82], [Dafermos 80], [Dantzig, et al, 79], [Feijoo and Meyer 84], [Lawphongpanich and Hearn 83], [Pang and Yu 84], hydroelectric power systems [Hanscom, et al, 80], and telecommunications networks [McCallum 76]. In many cases, the network and the number of the commodities are large, so such problems can have hundreds of thousands of variables. To solve MCP, several algorithms have been proposed. Among the best known are (1) the Frank-Wolfe approach [LeBlanc, et al, 75]; (2) the column generation approach [Leventhal, et al, 73]; and (3) the convex simplex approach [Nguyen 74].

Among these algorithms, only the Frank-Wolfe method leads to the decomposition of MCP into independent subproblems. The disadvantage of the Frank-Wolfe method is that it converges very slowly.

To accelerate convergence and to allow for parallelism, the approach of [Feijoo and Meyer 84] utilizes nonlinear separable approximation of the objective. Convex separable network flow problems have been successfully solved by many programming algorithms based on methods that iteratively generate search directions by solving linear subproblems. Five algorithms that have been used are the Frank-Wolfe [LeBlanc, et al, 75], [Collins, et al, 78], convex simplex [Nguyen 74], [Rosenthal 81], reduced gradient [Murtagh and Saunders 78], [Dembo and Klincewicz 81], [Beck, at el, 83], piecewise-linearization [Kao and Meyer 81], [Kamesam and Meyer 84], [Monma and Segal 82], and Newton methods [Klincewicz 83]. Convex piecewise-linear networks can be reformulated as linear networks, allowing solution by extremely fast algorithms.

The PL-approximation algorithm of [Kamesam and Meyer 84] is thus adopted in [Feijoo and Meyer 84] as a subroutine to solve the separable subproblems. It has worked successfully for small problems [Feijoo and Meyer 84,85], but for large-scale problems, some drawbacks appear. The computing time per major iteration increases rapidly after several iterations, indicating difficulty in dealing with the whole feasible set. To remedy this, we incorporate in the algorithm both the trust region theory developed in [Meyer 85] and a more flexible method of constructing separable approximations.

1.4 TRAFFIC ASSIGNMENT PROBLEMS

The assignment of traffic to a transportation network arises when urban traffic planners wish to estimate the flows that will result if the existing traffic network is modified. The areas linked by the traffic system are divided into zones. The directed graph composed of links and zones forms a transportation network. To model the equilibrium flow of traffic in a transportation network is to determine the number of

trips made between each pair of zones during a particular time of the day. The corresponding optimization principles are due to Wardrop [Wardrop 52]:
Wardrop's First Principle: The principle of equal travel times. That is, in an optimal assignment, the travel time between two points will be the same on all routes used and not greater than the travel time on any other path between the two nodes.
Wardrop's Second Principle: The principle of overall minimization. This states that the trips or movements are routed so that the sum of travel time for all the movements is a minimum. This is equivalent to minimizing the average travel time.

Wardrop's first principle has been shown to be the most relevant for urban traffic studies, whereas the second principle is applicable in a centrally controlled traffic system such as a railway. We give the mathematical formulation as follows.

Formulation

Consider a transportation network consisting of m nodes and n arcs. A node represents either a zone or an intersection of roads. A two-way traffic road arc is represented by two directed arcs of opposite directions. For convenience, it is assumed that nodes are numbered from 1 to m and arcs from 1 to n; the first $1, 2, , \cdots , k$ nodes are the origins. Let $t_j \geq 0$ be the total flow on arc j. Given the number of trips O_{qi} between origin-destination pair qi and n increasing arc delay functions $c_j(t_j)$, a traffic pattern satisfying one of Wardrop's principles is the optimal solution to the following problem:

$$\min \sum_{j=1}^{n} f_j(t_j) \qquad \text{(TAP)}$$

$$\sum_{j \varepsilon W_i} x_{qj} - \sum_{j \varepsilon V_i} x_{qj} = b_{qi} \ ,$$

$$(i=1, \cdots ,m, \ q=1, \cdots ,k \ )$$

$$x_{qj} \geq 0,$$

$$t_j = \sum_{q=1}^{k} x_{qj},$$

where

x_{qj} = flow from origin q on arc j;

W_i = set of arcs originating at node i;

V_i = set of arcs terminating at node i;

$b_{qi} = -O_{qi}$ if i is a destination zone

$= \sum_l O_{ql}$ if $i = q$

$= 0$ otherwise .

The corresponding objective functions are

$$f_j(t_j) = \int_0^{t_j} c_j(x)\, dx \quad (\textit{Wardrop's first principle})$$

$$= t_j \ c_j(t_j) \ . \ (\textit{Wardrop's second principle})$$

In traffic applications the functions c_j are convex and thus the convexity of f_j for non-negative flow is easily shown. For example, the delay function c_j used in a study of Winnipeg and Hull, Canada, is

$$c_j(z) = a_j(1+\alpha_j(\frac{z}{p_j})^{\beta_j}),$$

and that used in Sioux Falls, South Dakota by the U.S. Bureau of Public Roads is

$$c_j(z) = a_j(1+0.15(\frac{z}{p_j})^4),$$

where a_j is the travel time on arc j at mean free speed, α_j, β_j are parameters, and p_j is the designed capacity of arc j. If we use a node-arc formulation for the flow conservation and consider each origin as the source of a commodity, then the problem above is exactly an MCP discussed earlier in Chapter 2. If the objective function is linear, then TAP is equivalent to k one-source-to-all-node shortest problems. Thus, if we use linear approximation approach (e.g. the Frank-Wolfe algorithm), the linear cost c_j in each arc is nonnegative. By the nonnegativity, an efficient Dijkstra's algorithm[Dijkstra 59] takes $O(m^2)$ steps for a one-source-to-all-node shortest problem. Furthermore, if the network is sparse ($n << m^2$), Johnson-Dijkstra's algorithm[Johnson 73] takes only $O(n \log m)$.

CHAPTER 2

REVIEW OF PREVIOUS WORK

In this chapter we will review three well-known feasible direction algorithms that may be used in solving MCP. Let D denote diag(A, A, ..., A). A feasible direction algorithm has the general format:

Step 0: Obtain a feasible starting point $\mathbf{x}$

Step 1: Compute a search direction $\mathbf{p}$ such that $D\mathbf{p} = 0$ and $\nabla f \cdot \mathbf{p} < 0$.

Step 2: Determine a step length α^* to be the (approximate) solution to

$$\min_{\alpha > 0} f(\mathbf{x} + \alpha \mathbf{p})$$

Step 3: Set

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha^* \mathbf{p}$$

Step 4: If $\mathbf{x}$ satisfies some convergence criterion, stop. Otherwise, goto Step 1.

The primary difference among feasible direction algorithms lies in the manner in which the search direction p is calculated. In sections 2.1, 2.2, 2.3, we will discuss the following methods separately:

(1) the Frank-Wolfe method,

(2) the reduced gradient method, and

(3) the piecewise-linear method.

2.1 THE FRANK-WOLFE METHOD

The Frank-Wolfe method is the most straightforward approach for solving MCP. Given a feasible point at iteration i, say $\mathbf{x}^i$, we find another feasible point, say $\mathbf{y}^i$, by solving the linear program

$$\min \nabla f(\mathbf{x}^i)\cdot\mathbf{y}$$
$$\mathbf{D}\mathbf{y} = \mathbf{b}$$
$$\mathbf{y} \geq \mathbf{0}$$

where $\nabla f(\mathbf{x}^i)$ is the gradient of f evaluated at $\mathbf{x}^i$. LP can be easily decomposed into k small subproblems and can be solved in parallel. We then search the line segment between $\mathbf{x}^i$ and $\mathbf{y}^i$ for an improved solution $\mathbf{x}^{i+1}$, (i.e. $\mathbf{p} = \mathbf{y}^i - \mathbf{x}^i$ in Step 3) and the procedure is repeated. At each iteration, a lower bound on the optimal objective function value is calculated; termination occurs when difference between $f(\mathbf{x}^i)$ and this lower bound is smaller then a preassigned tolerance. The most appealing characteristic of the Frank-Wolfe method is that it can exploit parallelism to solve MCP, which has special constraint structure. Nonetheless, the main drawback is that it may experience slow convergence.

2.2 THE REDUCED GRADIENT METHOD

The reduced gradient method was first proposed by Wolfe[Wolfe 67], and was then elegantly coded as MINOS by Murtagh and Saunders[Murtagh and Saunders 78]. The decision variables are partitioned into three sets

$$\mathbf{x} = (\mathbf{x}_B, \mathbf{x}_S, \mathbf{x}_N);$$

where $\mathbf{x}_B$ are the basics, $\mathbf{x}_S$ are the superbasics, and $\mathbf{x}_N$ are fixed nonbasics. Each component of $\mathbf{x}_N$ is 0. The columns of D are similarly partitioned;

$$D = [B, S, N]$$

where B is square and nonsingular. Hence the basic variables may be expressed as a function of the superbasic and fixed nonbasic variables,

$$\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}S\mathbf{x}_S - B^{-1}N\mathbf{x}_N = \mathbf{y}(\mathbf{x}_S, \mathbf{x}_N) .$$

Then the objective function may be written as:

$$F(\mathbf{x}_S, \mathbf{x}_N) = f(\mathbf{y}(\mathbf{x}_S, \mathbf{x}_N), \mathbf{x}_S, \mathbf{x}_N) .$$

Temporarily ignoring the bounds on $\mathbf{x}_B$ and fixing $\mathbf{x}_N$, MCP may be replaced by the following reduced problem:

$$\min\ F(\mathbf{x}_S, \mathbf{x}_N)$$

$$\text{s.t. } \mathbf{x}_S \geq \mathbf{0} .$$

The gradient of F with respect to $\mathbf{x}_S$

$$\frac{\partial F}{\partial \mathbf{x}_S} = \frac{\partial f}{\partial \mathbf{x}_S} - \frac{\partial f}{\partial \mathbf{x}_B} B^{-1}S .$$

is referred to as the *reduced gradient.* It is computed as follows:

(1) compute the dual variables π by solving

$$\pi B = \frac{\partial f}{\partial \mathbf{x}_B} ,$$

(2) evaluate $\frac{\partial F}{\partial \mathbf{x}_S}$

$$\frac{\partial F}{\partial \mathbf{x}_S} = \frac{\partial f}{\partial \mathbf{x}_S} - \pi S .$$

Algorithm

Let $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_B, \bar{\mathbf{x}}_S, \bar{\mathbf{x}}_N)$ denote the current values of $\mathbf{x}$.

1. Evaluate $\frac{\partial F}{\partial \mathbf{x}_S}$ at $\bar{\mathbf{x}}$.
2. If certain optimality tests on the reduced problem, using the current set of tolerances, are met, continue. Otherwise, goto step 7.
3. Compute $\frac{\partial F}{\partial \mathbf{x}_N}$ using a formula analogous to $\frac{\partial F}{\partial \mathbf{x}_S}$.

4. If $\frac{\partial F}{\partial \mathbf{x}_N} \geq \mathbf{0}$, continue. Otherwise, goto step 6.
5. If the optimality tolerances on the reduced problem are "tight", stop. Otherwise, replace the current loose tolerances by the tight ones and goto step 7.
6. Add one or more nonbasic variables to the superbasic set.
7. Calculate the search direction vector $\mathbf{d}_S$ and maximum stepsize α_S for the superbasic variables. The scalar α_S is the largest value of α such that $\overline{\mathbf{x}}_S + \alpha\mathbf{d}_S$ satisfies the bounds on $\mathbf{x}_S$.
8. Calculate $\mathbf{d}_B$, a search direction vector, and maximum stepsize α_B for the basic variables. The vector $\mathbf{d}_B$ is computed by solving:
$$\mathbf{B}\mathbf{d}_B = -\mathbf{S}\mathbf{d}_S.$$
9. (Line Search). Let $\bar{\alpha} = \min\{\alpha_S, \alpha_B\}$ and $\mathbf{d} = (\mathbf{d}_B, \mathbf{d}_S, \mathbf{0})$. Find an approximate solution, α^*, to the one dimensional optimization problem:
$$\min f(\overline{\mathbf{x}} + \alpha\mathbf{d})$$
$$\text{s.t. } 0 \leq \alpha \leq \bar{\alpha}$$
10. Replace $\mathbf{x}$ by $\overline{\mathbf{x}} + \alpha\mathbf{d}$. If $\alpha^* < \bar{\alpha}$ goto step 1.
11. If a basic variable becomes 0, make it nonbasic and replace it with a superbasic. Goto step 1.

The search direction $\mathbf{d}_S$ in step 7 can be obtained either by the quasi-Newton algorithm or by the conjugate gradient (CG) algorithm. Both methods require only first partial derivatives of the objective and achieve rapid convergence. However, the quasi-Newton algorithm demands too much storage, which is a great drawback especially in cases of large-scale problems.

2.3 THE PIECEWISE-LINEAR METHOD

The piecewise-linear method for MCP was proposed in [Feijoo and Meyer 84]. Basically, our method in this thesis is an extension of this method.

Algorithm

1. Find a separable approximation $f^S(\mathbf{x})$ at current feasible solution $\overline{\mathbf{x}}$.

   A separable function $f^S(\mathbf{x})$, which replicates $f(\mathbf{x})$ if only a single element of $\mathbf{x}$ is changed, is used to approximate $f(\mathbf{x})$. That is

   $$f^S(\mathbf{x}) = f(\overline{\mathbf{x}}) + \sum_{j=1}^{n} [\, f(\,..\,, \overline{x}_{j-2}, \overline{x}_{j-1}, x_j, \overline{x}_{j+1}, \overline{x}_{j+2}, \,..\,) - f(\overline{\mathbf{x}})\,]$$

   $$\text{where } \mathbf{x} = (x_1, x_2, \ldots, x_n)$$

   (In solving MCP, the resulting separable programming problem can be decomposed into k small subproblems and can be solved in parallel.)
2. Generate a separable piecewise-linear approximation $f^{PL}(\mathbf{x})$(see Fig. 2.1).
3. Solve the implicit linear program to get a new solution point $\mathbf{y}$.

   The approximating piecewise-linear program can be reformulated as a linear program by replacing each arc by as many arcs as the number of the piecewise segments(see Fig. 2.1). By the method proposed by [Kamesam and Meyer 84], it can be solved implicitly by a 2n-dimension linear program.
4. Do line-search for $\lambda > 0$ on $\overline{\mathbf{x}} + \lambda \cdot (\mathbf{y} - \overline{\mathbf{x}})$.

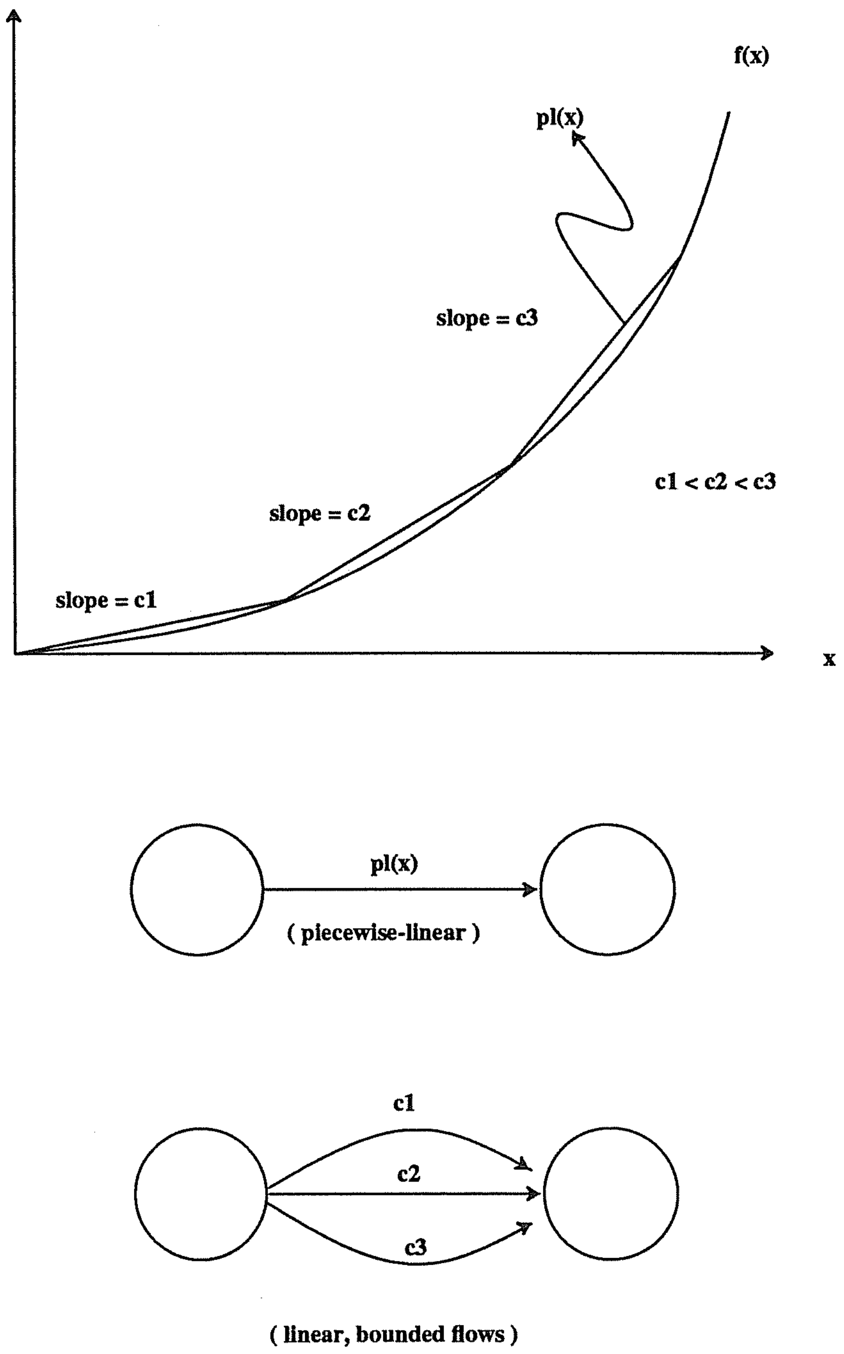


Figure 2.1 Piecewise Linear Approximation Method

CHAPTER 3

SCALED SEPARABLE APPROXIMATIONS

3.1 INTRODUCTION

To obtain a good objective function approximation, we consider a class of scaled separable approximations in this chapter. This scaling has the property that the true objective function is bounded below and above by separable functions corresponding to appropriate choices of the scaling parameters. Letting f be convex and differentiable, for a feasible point $\mathbf{x} \in \Omega$, we define a shifted function on R^{kn}

$$h(\mathbf{d}) := f(\mathbf{d}+\mathbf{x}) - f(\mathbf{x}).$$

This function corresponds to the change in the objective function resulting from a change of **d** in the current flow **x**. Thus, MCP is equivalent to

$$\min h(\mathbf{d}) \;\; s.t. \;\; \mathbf{d} \in \Omega_{\mathbf{x}} := \{ \mathbf{d} \mid \mathbf{x}+\mathbf{d} \in \Omega \}. \qquad (\text{MCP}_1)$$

For consistency , we list the notation associated with h in the same manner as before:

Notation

$\mathbf{d}_q := (d_{q1}, d_{q2}, \cdots , d_{qn})$ (flow vector for changes in commodity q);

$\mathbf{d}_{.j} := (d_{1j}, d_{2j}, \cdots , d_{kj})$ (flow vector of changes in arc j);

$\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \cdots , \mathbf{d}_k)$ (full set of flow vector changes);

$h_j(\cdot) := f_j(\cdot + \sum_q x_{qj}) - f_j(\sum_q x_{qj})$ (shifted cost function for arc j).

Definition

For a $\sigma > 0$, a *scaled separable function of h* with scale factor σ is defined by

$$h^S(\mathbf{d},\sigma) := \sum_j \sum_q \frac{1}{\sigma} h_j(\sigma d_{qj}).$$

Some simple examples of scaled separable approximations are given below. The upper and lower bounds for h are from corollary 3.4 in section 3.3.

3.2 EXAMPLES OF SCALED SEPARABLE APPROXIMATIONS

Example 1 $f(x) = x^2 \quad (k = 1, n = 1)$

[1] $\bar{x} = 0$: ($\bar{x}$ denotes current point)

$h(d) = h_1(d) = d^2$

$h^S(d,\sigma) = \frac{1}{\sigma}(\sigma d)^2 = \sigma d^2$

(1) $\sigma = 1$ (*upper bound*)

$h^S(d,1) = d^2 = h(d)$

(2) $\sigma = 0$ ($h^S(d,0) := h'(0)\cdot d$) (*lower bound*)

$h^S(d,0) = 0$

[2] $\bar{x} = 1$:

$h(d) = h_1(d) = (1+d)^2 - 1 = 2d + d^2$

$h^S(d,\sigma) = \frac{1}{\sigma}(2(\sigma d) + (\sigma d)^2) = 2d + \sigma d^2$

(1) $\sigma = 1$ (*upper bound*)

$h^S(d,1) = 2d + d^2 = h(d)$

(2) $\sigma = 0$ (*lower bound*)

$h^S(d,0) = 2d \ (= f'(1)\cdot d)$

Example 2 $f(x,y) = (x+y)^2 \quad (k = 2, n = 1)$

[1] $(\bar{x},\bar{y}) = (0,0)$: ($(\bar{x},\bar{y})$ denotes current point)

$h(\mathbf{d}) = h_1(d_{11}+d_{21}) = (d_{11}+d_{21})^2$ (see Fig. 3.1)

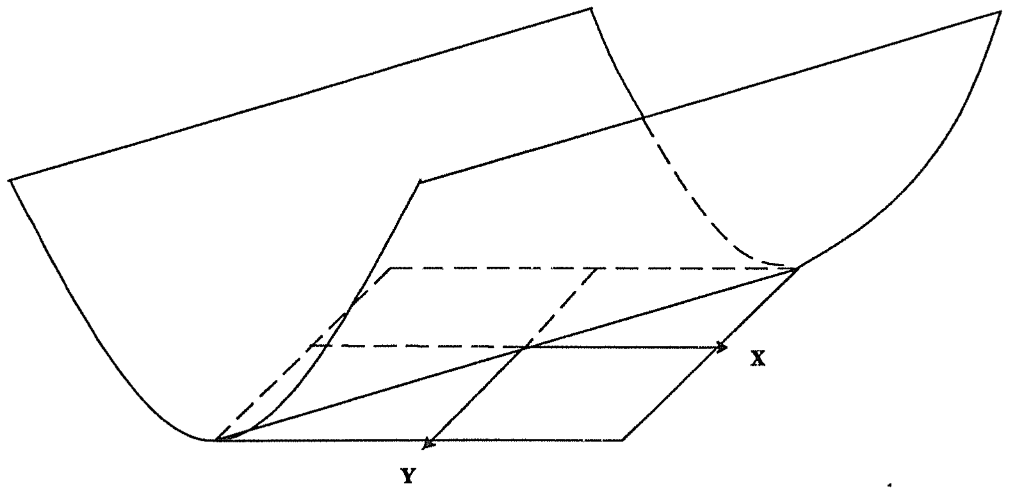


Figure 3.1 $h(x,y) = (x+y)^2$

$$h^S(\mathbf{d},\sigma) = \frac{1}{\sigma}(\sigma d_{11})^2 + \frac{1}{\sigma}(\sigma d_{21})^2 = \sigma(d_{11}^2 + d_{21}^2)$$

(1) $\sigma = 2$ (*upper bound*) (see Fig. 3.2)

$h^S(d,2) = 2(d_{11}^2 + d_{21}^2)$

(2) $\sigma = 0$ (*lower bound*)

$h^S(\mathbf{d},0) = 0$ ($= \nabla f(\mathbf{0}) \cdot \mathbf{d}$)

(3) $\sigma = 1$

$h^S(d,1) = d_{11}^2 + d_{21}^2$

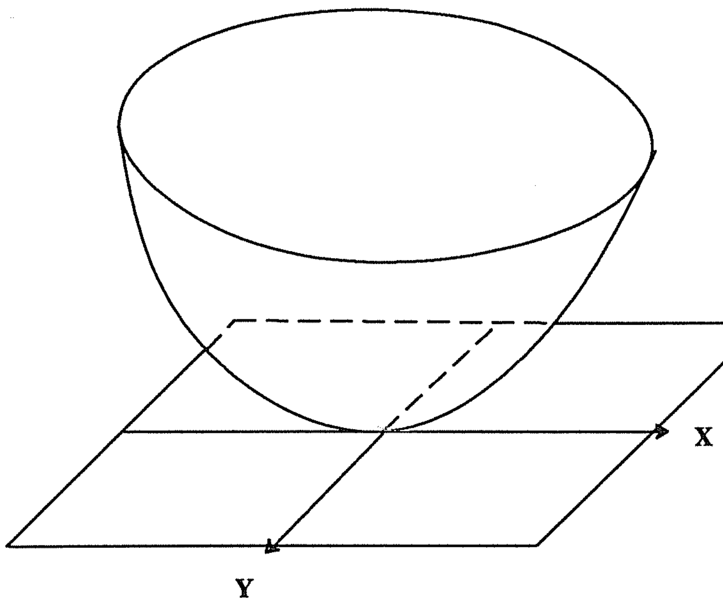


Figure 3.2 $h(x,y) = 2x^2 + 2y^2$

[2] $(\bar{x},\bar{y}) = (1,1)$:

$h(\mathbf{d}) = h_j(d_{11}+d_{21}) = (1+d_{11}+1+d_{21})^2-(1+1)^2 = (d_{11}+d_{21})^2+4(d_{11}+d_{21})$

$h^S(\mathbf{d},\sigma) = \frac{1}{\sigma}((\sigma d_{11})^2+4(\sigma d_{11}))+\frac{1}{\sigma}((\sigma d_{21})^2+4(\sigma d_{21})) = \sigma(d_{11}^2+d_{21}^2)+4(d_{11}+d_{21})$

(1) $\sigma = 2$ (*upper bound*)

$h^S(d,2) = 2(d_{11}^2+d_{21}^2)+4(d_{11}+d_{21})$

(2) $\sigma = 0$ (*lower bound*)

$h^S(\mathbf{d},0) = 4(d_{11}+d_{21})$ ($= \nabla f(1,1)\cdot\mathbf{d}$)

Precision

The error function in example 2 is $\sigma(d_{11}^2+d_{21}^2)-(d_{11}+d_{21})^2$(independent of the base point). The contours of error functions for $\sigma = 0$, $\sigma = 1$, and $\sigma = 2$ are shown in Figure 3.3, 3.4, and 3.5.

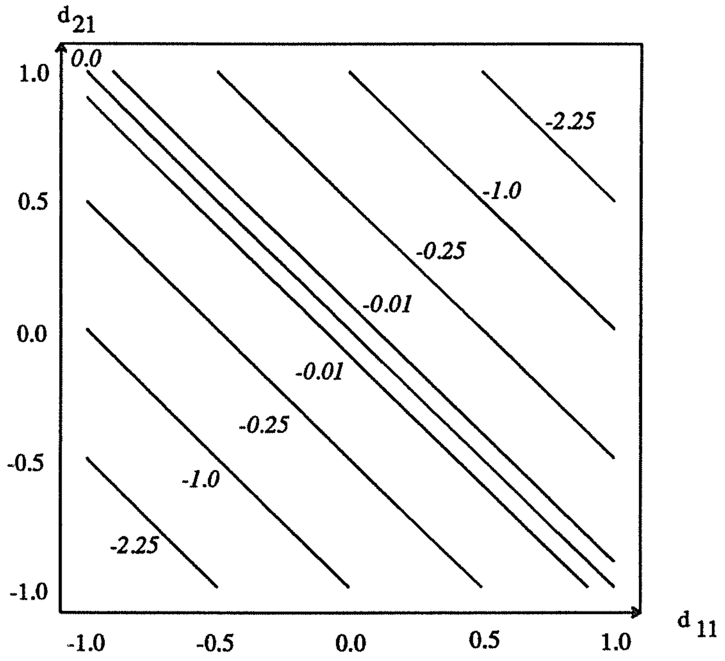


Figure 3.3 Error Contours for σ = 0

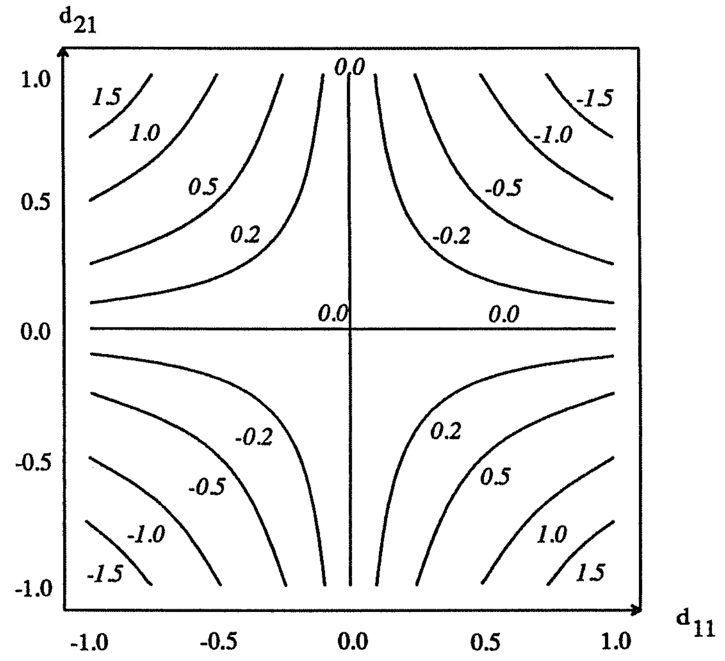


Figure 3.4 Error Contours for σ = 1

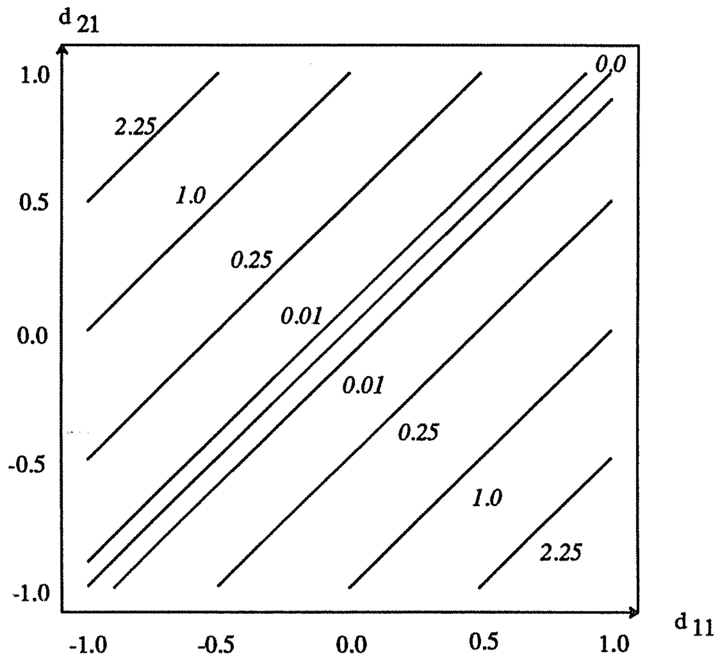


Figure 3.5 Error Contours for $\sigma = 2$

3.3 PROPERTIES OF SCALED SEPARABLE FUNCTIONS

Observe that h_j and $h(\mathbf{d}) = \sum_j h_j(\sum_q d_{qj})$ are convex, and $h_j(0) = 0$ for $j = 1, 2, \cdots, n$. By these properties, we have the following lemma.

Lemma 3.1 $h(\mathbf{d}) \le h^S(\mathbf{d},k)$.

Proof: By the convexity of h_j, and $\sum_{q=1}^{k} \frac{1}{k} = 1$, we have

$$h_j(\sum_{q=1}^{k} d_{qj}) = h_j(\sum_{q=1}^{k} \frac{1}{k} \cdot kd_{qj}) \le \sum_{q=1}^{k} \frac{1}{k} \cdot h_j(kd_{qj})$$

The lemma follows by applying $\sum_{j=1}^{n}$ on both sides. $\square$

We now show the monotonicity of $h^S(\mathbf{d},\cdot)$.

Lemma 3.2 For $0 < \sigma_1 < \sigma_2$, $h^S(\mathbf{d},\sigma_1) \le h^S(\mathbf{d},\sigma_2)$.

Proof: By the convexity of h_j, and $h_j(0) = 0$, we have

$$h_j(\sigma_1 d_{qj}) = h_j\left(\frac{\sigma_1}{\sigma_2} \cdot \sigma_2 d_{qj} + \frac{\sigma_2 - \sigma_1}{\sigma_2} \cdot \sigma_2 \cdot 0 \right) \le \frac{\sigma_1}{\sigma_2} \cdot h_j(\sigma_2 d_{qj})$$

The lemma follows by applying $\sum_{j=1}^{n} \sum_{q=1}^{k} \sigma_1^{-1}$ on both sides. □

The following lemma shows that the limit as $\sigma \to 0$ of the separable functions $h^S(\mathbf{d},\sigma)$ is the linearization approximation at 0.

Lemma 3.3 For fixed $\mathbf{d}$, $h^S(\mathbf{d},\sigma) \to \nabla h(\mathbf{0})\mathbf{d} = \nabla f(\mathbf{x})\mathbf{d}$, as $\sigma \to 0^+$.

Proof: By L'Hospital's Rule, we have

$$\lim_{\sigma \to 0^+} \frac{h_j(\sigma d_{qj})}{\sigma} = \lim_{\sigma \to 0^+} h_j'(\sigma d_{qj}) \cdot d_{qj} = h_j'(0) \cdot d_{qj};$$

and by the definition of h_j,

$$\frac{\partial h(\mathbf{0})}{\partial d_{qj}} = h_j'(0) = f_j'(\textstyle\sum_q x_{qj}) = \frac{\partial f(\mathbf{x})}{\partial x_{qj}}. \quad \square$$

Accordingly, if we define $h^S(\mathbf{d},0) := \nabla h(\mathbf{0})\mathbf{d}$, then $h^S(\mathbf{d},\cdot)$ is an increasing continuous function on $[0,\infty)$, and the next corollary notes that the choices $\sigma = 0$, $\sigma = k$ yield lower and upper bounds respectively on the objective function.

Corollary 3.4 $h^S(\mathbf{d},0) \le h(\mathbf{d}) \le h^S(\mathbf{d},k)$.

Thus, $h^S(\mathbf{d},k)$ serves as an upper bound of $h(\mathbf{d})$, while $h^S(\mathbf{d},0)$ serves as a lower bound of $h(\mathbf{d})$. To construct a good separable approximation of the objective is a key factor in the efficiency of the trust region method. The continuity and monotonicity of $h^S(\mathbf{d},\cdot)$ indicate that with a proper σ, $h^S(\mathbf{d},\sigma)$ is a suitable approximation. We allow the scalar σ to be adjusted at every iteration of our algorithm. The amount of increase or decrease of σ is determined by the ratio of the true and approximate objective function improvements in the preceding iteration.

The next lemma is used to show that the direction obtained from the separable program (in which the scaled separable approximate function replaces the objective) is a descent direction.

Lemma 3.5 For $\sigma \geq 0$, $\nabla h^S(\mathbf{0},\sigma) = \nabla h(\mathbf{0})$.

Proof: For $\sigma = 0$, the result is trivial.

For $\sigma > 0$,

$$\frac{\partial h^S(\mathbf{d},\sigma)}{\partial d_{qj}} = \frac{\partial}{\partial d_{qj}}\left(\frac{1}{\sigma} h_j(\sigma d_{qj})\right) = h_j'(\sigma d_{qj}).$$

The lemma then follows from

$$\frac{\partial h^S(\mathbf{0},\sigma)}{\partial d_{qj}} = h_j'(0) = \frac{\partial h(\mathbf{0})}{\partial d_{qj}}. \quad \square$$

Assuming a σ has been selected, we consider an approximating separable program

$$\min h^S(\mathbf{d},\sigma) \ \ s.t. \ \ \mathbf{d} \in \Omega_{\mathbf{x}} \qquad \text{(SP)}$$

If we define $h_q^S(\mathbf{d}_q,\sigma) := \sum_{j=1}^{n} h_{qj}^S(d_{qj},\sigma)$, then the objective $h^S(\mathbf{d},\sigma) = \sum_{q=1}^{k} h_q^S(\mathbf{d}_q,\sigma)$. The separable program then can be decomposed into k subprograms of the form:

$$\min h_q^S(\mathbf{d}_q,\sigma) \ \ s.t. \ \ \mathbf{d}_q \in \Omega_{\mathbf{x}_q} := \{ \mathbf{d}_q \mid \mathbf{x}_q + \mathbf{d}_q \in \Omega_q \} \qquad (\text{SP}_q)$$

The next lemma establishes the descent relation between f and its separable approximations.

Lemma 3.6 For $\sigma \geq 0$, if $h^S(\mathbf{d},\sigma) < 0$, then $\mathbf{d}$ is a descent direction of f at $\mathbf{x}$.

Proof: For $\sigma = 0$, the result is trivial.

For $\sigma > 0$, by the convexity of $h^S(\cdot,\sigma)$ and $h^S(\mathbf{0},\sigma) = 0$, we have

$$0 > h^S(\mathbf{d},\sigma) \geq h^S(\mathbf{0},\sigma) + \nabla h^S(\mathbf{0},\sigma)\cdot\mathbf{d} = \nabla h(\mathbf{0})\cdot\mathbf{d} = \nabla f(\mathbf{x})\cdot\mathbf{d}$$

So $\mathbf{d}$ is a descent direction of f at $\mathbf{x}$. $\square$

The final step in the approximation process is to replace h^S by a piecewise-linear approximation h^{PL}. Specifically, let h^{PL} be the piecewise-linear approximation of h^S with fixed mesh-size $\delta > 0$. That is,

$$h^{PL}(\mathbf{d},\sigma) := \sum_j \sum_q \frac{1}{\sigma} h_j^{PL}(\sigma d_{qj}) ,$$

where the functions $h_j^{PL}(\sigma d_{qj})$ are piecewise-linear functions that agree with $h_j(\sigma d_{qj})$ at a set of mesh points to be described below. We define PLP and PLP$_q$ analogous to SP and SP$_q$:

$$\min h^{PL}(\mathbf{d},\sigma) \ \ s.t. \ \ \mathbf{d} \ \varepsilon \ \Omega_{\mathbf{x}} \qquad \text{(PLP)}$$

$$\min h_q^{PL}(\mathbf{d}_q,\sigma) \ \ s.t. \ \ \mathbf{d}_q \ \varepsilon \ \Omega_{\mathbf{x}_q} . \qquad (\text{PLP}_q)$$

Moreover, these subproblems can be solved in parallel [Feijoo and Meyer 85]. If the trust region contains the original feasible set, then the method of Feijoo and Meyer corresponds to $\sigma = 1$, while the Frank-Wolfe method [LeBlanc, et al., 75] corresponds to $\sigma = 0$.

Since $h^{PL} \geq h^S$, the preceding lemma can be used to show that a descent direction is obtained if the optimal value of PLP is less than zero.

Lemma 3.7 For $\sigma \geq 0$, if $h^{PL}(\mathbf{d},\sigma) < 0$, then $\mathbf{d}$ is a descent direction of f at $\mathbf{x}$.

In the algorithms below, we use h^{PL} as the approximating function. This approximation has the advantage that the resulting subproblems may be solved as linear programs.

3.4 EXTENSION TO GENERAL LINEAR CASE

In this section we will extend the scaled separable approximations to a more general class of nonseparable functions. Let

$$g(\mathbf{d}) := g_1(\mathbf{c}\cdot\mathbf{d}) = g_1(\sum_{q=1}^{K} c_q d_q),$$

where $\mathbf{d}, \mathbf{c} \in R^K$, $g_1 : R \to R$ is convex with $g_1(0) = 0$, and $g : R^K \to R$.

Definition

For a $\sigma > 0$, a *scaled separable function of g* with scale factor σ is defined by

$$g^S(\mathbf{d},\sigma) := \sum_{q=1}^{K} \frac{1}{\sigma} g_j(\sigma c_q d_q).$$

Similar to the foregoing section, we have following lemmas.

Lemma 3.8 $g(\mathbf{d}) \le g^S(\mathbf{d},K_1)$, where K_1 is the number of non-zeros in $\mathbf{c}$.

Proof: By the convexity of g_1, and $\sum_{q=1}^{K_1} \frac{1}{K_1} = 1$, we have

$$g_1(\sum_1^K c_q d_q) = g_1(\sum_{c_q \neq 0} c_q d_q) = g_1(\sum_{c_q \neq 0} \frac{1}{K_1}\cdot K_1 c_q d_q)$$

$$\le \sum_{c_q \neq 0} \frac{1}{K_1}\cdot g_1(K_1 c_q d_q) = \sum_1^K \frac{1}{K_1}\cdot g_1(K_1 c_q d_q). \quad \square$$

Lemma 3.9 For $0 < \sigma_1 < \sigma_2$, $g^S(\mathbf{d},\sigma_1) \le g^S(\mathbf{d},\sigma_2)$.

Proof: By the convexity of g_1, and $g_1(0) = 0$, we have

$$g_1(\sigma_1 c_q d_q) = g_1(\frac{\sigma_1}{\sigma_2}\cdot\sigma_2 c_q d_q + \frac{\sigma_2-\sigma_1}{\sigma_2}\cdot\sigma_2\cdot 0) \le \frac{\sigma_1}{\sigma_2}\cdot g_1(\sigma_2 c_q d_q)$$

The lemma follows by applying $\sum_{q=1}^{K_1} \sigma_1^{-1}$ on both sides. $\square$

Lemma 3.10 For fixed $\mathbf{d}$, $g^S(\mathbf{d},\sigma) \to \nabla g(\mathbf{0})\mathbf{d}$ as $\sigma \to 0^+$.

Proof: By L'Hospital's Rule, we have

$$\lim_{\sigma\to 0^+} \frac{g_1(\sigma c_q d_q)}{\sigma} = \lim_{\sigma\to 0^+} g_1'(\sigma c_q d_q)\cdot c_q d_q = g_1'(0)\cdot c_q d_q;$$

and the lemma follows by the definition of ∇g

$$\frac{\partial g(\mathbf{0})}{\partial d_q} = g_1'(0)\cdot c_q. \quad \square$$

We now define $g^S(\mathbf{d},0) := \nabla g(\mathbf{0})\mathbf{d}$.

Corollary 3.11 $g^S(\mathbf{d},0) \le g(\mathbf{d}) \le g^S(\mathbf{d},K_1)$.

Lemma 3.12 For $\sigma \ge 0$, $\nabla g^S(\mathbf{0},\sigma) = \nabla g(\mathbf{0})$.

Proof: For $\sigma = 0$, the result is trivial.

For $\sigma > 0$,

$$\frac{\partial g^S(\mathbf{d},\sigma)}{\partial d_q} = \frac{\partial}{\partial d_q}\left(\frac{1}{\sigma} g_1(\sigma c_q d_q)\right) = g_1'(\sigma c_q d_q)\cdot c_q.$$

The lemma then follows from

$$\frac{\partial g^S(\mathbf{0},\sigma)}{\partial d_q} = g_1'(0)\cdot c_q = \frac{\partial g(\mathbf{0})}{\partial d_q}. \quad \square$$

Lemma 3.13 For $\sigma \ge 0$, if $g^S(\mathbf{d},\sigma) < 0$, then $\mathbf{d}$ is a descent direction of g at $\mathbf{0}$.

Proof: For $\sigma = 0$, the result is trivial.

For $\sigma > 0$, by the convexity of $g^S(\cdot,\sigma)$ and $g^S(\mathbf{0},\sigma) = 0$, we have

$$0 > g^S(\mathbf{d},\sigma) \ge g^S(\mathbf{0},\sigma) + \nabla g^S(\mathbf{0},\sigma)\cdot\mathbf{d} = \nabla g(\mathbf{0})\cdot\mathbf{d}$$

So $\mathbf{d}$ is a descent direction of g at $\mathbf{0}$. $\square$

The results in this section can be extended to the nonseparable functions of the form

$$g(\mathbf{d}) := \sum_{j=1}^{J} g_j(\mathbf{c}_j \cdot \mathbf{d}),$$

where

$\mathbf{d}, \mathbf{c}_j \, \varepsilon R^K$ for $j = 1, \cdots, J$,

$g_j : R \rightarrow R$ is convex with $g_j(0) = 0$ for $j = 1, \cdots, J$, and

$g : R^K \rightarrow R$.

The function class in the previous section is a special case when $K = k, J = n$ and

$c_1 = (1, 0, .., 0, 1, 0, .., 0, 1, 0, .., 0)$

$c_2 = (0, 1, .., 0, 0, 1, .., 0, 0, 1, .., 0)$

.

.

$c_J = (0, 0, .., 1, 0, 0, .., 1, 0, 0, .., 1)$.

Results analogous to lemmas 3.8-3.12 are then easily established for $g(\mathbf{d})$. For each term g_j the value of σ needed to guarantee an upper bound is the number of non-zeros in $\mathbf{c}_j$.

CHAPTER 4

TRUST REGION METHODS

4.1 INTRODUCTION

The trust region technique for unconstrained nonlinear programs was first described in [Moré and Sorensen 79], [Fletcher 81], [Sorensen 82] and was further developed by a few other authors to solve constrained problems: [Zhang, et al, 84], [Vardi 85], [Meyer 85]. In this chapter we will discuss two types of trust region methods for a general nonlinear program with linear constraints as follows:

$$\min f(\mathbf{x}) \qquad \text{(NLP)}$$
$$s.t.\ A\mathbf{x} \leq \mathbf{b},$$

where we assume that f is a continuously differentiable convex function on the feasible set $X := \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\} \subset R^n$ and A is an $m \times n$ matrix.

A trust region means a proper neighborhood in which an approximating objective function is defined so that the original objective function is not too far from the approximating one. Discussed in the following two sections are a linear trust region algorithm, presented in [Zhang 84], and a piecewise-linear trust region algorithm, proposed in [Meyer 85]. The approximating function in section 4.2 is linear, while in section 4.3 it is piecewise-linear. Because both methods are of iterated linear program type, they are suitable to tackle very large-scale problems. From the theory discussed in section 4.3, we develop in the next chapter an algorithm to solve MCP, a special case of NLP.

4.2 A LINEAR TRUST REGION ALGORITHM

The linear trust region algorithm(LTR) has a prototype just as the Frank-Wolfe algorithm does in Chapter 2. That is, at each iteration we solve a linear program with linear approximation objective function. Moreover, in LTR a cubic constraint is imposed on the original constraints to form a trust region at each iteration. Unlike Frank-Wolfe algorithm, LTR does not do a line-search but just decides whether to accept the solution point or to reject it. The choice of neighborhood and the acceptance or rejection of a trial point are determined by the ratio of the improvement in the original objective function over the improvement in the linear approximation function.

Given a feasible point $\mathbf{x}$ in NLP, we define a "shifted" function $h(\mathbf{d})$, where $\mathbf{d}$ denotes the difference between $\mathbf{x}$ and its potential successor and h is the difference in their function values:

$$h(\mathbf{d}) := f(\mathbf{d}+\mathbf{x}) - f(\mathbf{x}).$$

Note that $\mathbf{d} = \mathbf{0}$ thus corresponds to $\mathbf{x}$, with $h(\mathbf{0}) = 0$, and $h(\mathbf{d}) < 0$ if $\mathbf{d} + \mathbf{x}$ has a lower objective function value than f.

At each iteration of the algorithm to be defined below, we solve a linear approximating problem LP(α,x):

$$\min_{\mathbf{d}} h^L(\mathbf{d}) := \nabla h(\mathbf{0})\cdot\mathbf{d}$$

$$s.t.\ \ \mathrm{A}\mathbf{d} \le \bar{\mathbf{b}}, \quad \|\mathbf{d}\| \le \alpha,$$

where α is the *trust region parameter*, $\bar{\mathbf{b}} := \mathbf{b} - \mathrm{A}\mathbf{x}$, and $\|\mathbf{d}\|$ represents the l_∞ norm. Observe that $\mathrm{A}\mathbf{d} \le \bar{\mathbf{b}}$ is equivalent to $\mathrm{A}(\mathbf{x} + \mathbf{d}) \le \mathbf{b}$. $\mathbf{d}^L(\alpha,\mathbf{x})$ will denote any solution of LP(α,x). We will omit argument $\mathbf{x}$ if not ambiguous.

To know when to stop the algorithm, we need to develop optimality conditions. The following lemma states first-order optimality conditions, involving trust region, for NLP.

Lemma 4.1 **x** is an optimal solution of NLP if and only if $\mathbf{d}^L = \mathbf{0}$ solves LP(α,**x**) for $\alpha > 0$.

Proof: Suppose LP(α,**x**) solved by $\mathbf{d} = \mathbf{0}$

=> KKT for LP(α,**x**) satisfied at $\mathbf{d} = \mathbf{0}$

=> KKT for NLP(**x**) satisfied at $\mathbf{d} = \mathbf{0}$

=> $\mathbf{d} = \mathbf{0}$ is optimal for NLP because of convexity of h(**d**).

Analogous argument holds in other direction. □

We now state the algorithm:

LTR Algorithm

step 0:

Select positive α^0, and $\underline{\alpha}$, scalars $0 < \rho_0 < \rho_1 < \rho_2 < 1$ and $\beta > 1$.

Set $i = 0$.

step 1:

Find $\mathbf{x}^0$ satisfying $A\mathbf{x} \leq \mathbf{b}$.

step 2:

Solve LP(α^i,$\mathbf{x}^i$), obtaining an optimal solution $\mathbf{d}^{iL}$.

step 3:

Compute the actual change $h^i := h(\mathbf{d}^{iL})$ and predicted change $h^{iL} := h^L(\mathbf{d}^{iL})$.

If $h^{iL} = 0$, stop.

Otherwise compute the ratio of actual to predicted change

$$\rho^i := \rho(\alpha^i, \mathbf{x}^i) = \frac{h^i}{h^{iL}}.$$

step 4: (see Fig. 4.1)

If $\rho^i < \rho_0$, then $\alpha^i \leftarrow \frac{\alpha^i}{\beta}$, goto step 2; (case 4) otherwise take $\mathbf{x}^{i+1} \leftarrow \mathbf{x}^{i}$ and update α^{i+1} by

$$\alpha^{i+1} = \begin{cases} \max\left\{ \dfrac{\alpha^i}{\beta}, \underline{\alpha} \right\} & \text{if } \rho^i \leq \rho_1; \text{ (case3)} \\ \max\left\{ \beta\alpha^i, \underline{\alpha} \right\} & \text{if } \rho^i \geq \rho_2; \text{ (case1)} \\ \max\left\{ \alpha^i, \underline{\alpha} \right\} & \text{otherwise. (case2)} \end{cases}$$

$i \leftarrow i+1$ and goto step 2.

In the algorithm, the trust region is defined by the constraint $\|\mathbf{d}\| \leq \alpha$, and the ratio ρ is used to judge if α is of proper size. A good test is to check if the predicted and actual changes at optimal solution of LP are close. If ρ is above the upper limit ρ_2, size of trust region α is multiplied by β; and if ρ is below the lower limit ρ_1, α is divided by β. We can choose $\rho_2 = 0.9$ to 0.95 because ρ is no more than one by the convexity of f. A key point to guarantee the convergence in the trust region methods is that the initial value of the trust region size for each distinct $\mathbf{x}^i$ is at least $\underline{\alpha}$.

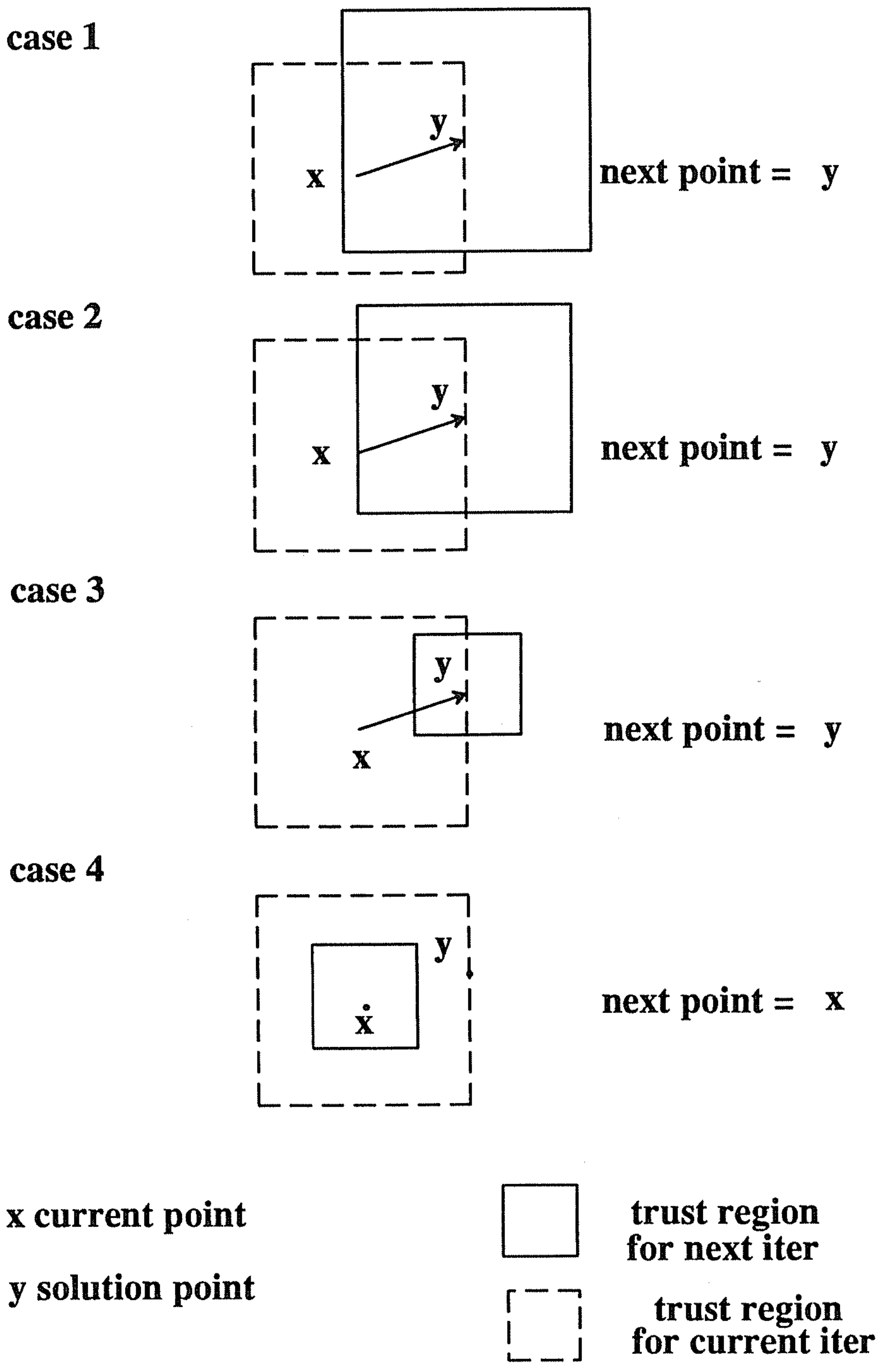


Figure 4.1 The Linear Trust Region Scheme

Before we prove the convergence theorem, we need some lemmas.

Lemma 4.2 For $\lambda \in [0,1]$, $h^L(\mathbf{d}^L(\lambda\alpha)) \leq h^L(\lambda\mathbf{d}^L(\alpha)) = \lambda h^L(\mathbf{d}^L(\alpha))$.

Proof: The left inequality for h^L follows from the feasibility of $\lambda\mathbf{d}^L(\alpha)$ for the trust region corresponding to $\lambda\alpha$. The right equality follows from $\mathbf{d}^L(0) = \mathbf{0}$ and the linearity of h^L. □

Lemma 4.3 If for some $\tilde{\alpha}$, $h^L(\mathbf{d}^L(\tilde{\alpha})) < 0$, then for any $\rho_0 < 1$, the inequality $\rho(\lambda\tilde{\alpha}) \geq \rho_0$ holds for all sufficiently small $\lambda > 0$.

Proof:

$$\begin{aligned}\rho(\lambda\tilde{\alpha}) &= \frac{h(\mathbf{d}^L(\lambda\tilde{\alpha}))}{h^L(\mathbf{d}^L(\lambda\tilde{\alpha}))}\\ &= \frac{h^L(\mathbf{d}^L(\lambda\tilde{\alpha})) + o(\lambda\tilde{\alpha})}{h^L(\mathbf{d}^L(\lambda\tilde{\alpha}))}\\ &= 1 + \frac{o(\lambda\tilde{\alpha})}{h^L(\mathbf{d}^L(\lambda\tilde{\alpha}))}\\ &= 1 + \frac{o(\lambda\tilde{\alpha})}{\lambda h^L(\mathbf{d}^L(\tilde{\alpha}))}.\end{aligned}$$

The result follows by noting that the last ratio tends to 0 as $\lambda \to 0$. □

In order to guarantee that the improvement ratio behaves properly in the neighborhood of **x**, we now prove that $h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x})$ is continuous.

Lemma 4.4 $h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x})$ is a continuous function of **x** and α for $\mathbf{x} \in X$ and $\alpha > 0$.

Proof: Suppose $(\mathbf{y}^i,\alpha^i) \to (\mathbf{x},\alpha)$ where $y^i \in X$ and $\alpha^i \geq 0$, and let h_i^{*L} denote the optimal value of the problem corresponding to $(\mathbf{y}^i,\alpha^i)$. By considering an appropriate convergent subsequence of optimal solutions it follows that

$$\liminf h_i^{*L} \geq h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x}).$$

To establish the other required inequality,

$$\limsup h_i^{*L} \le h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x}),$$

we construct a sequence of feasible solutions for the sequence of problems LP($\mathbf{y}^i,\alpha^i$) as follows: let $\mathbf{z} := \mathbf{x} + \mathbf{d}^L(\alpha,\mathbf{x})$, where $\mathbf{d}^L(\alpha,\mathbf{x})$ is any optimal solution of LP($\mathbf{x},\alpha$), (we assume $\mathbf{d}^L(\alpha,\mathbf{x}) \ne \mathbf{0}$, since the result is trivial if $h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x}) = 0$) and define λ_i and $\mathbf{d}^i$ such that $\mathbf{y}^i + \mathbf{d}^i = \mathbf{z}$ and such that λ_i is the largest scalar in [0,1] such that $\|\lambda_i \mathbf{d}^i\| \le \alpha^i$. Since $\mathbf{d}^i \to \mathbf{d}^L(\alpha,\mathbf{x})$ and $\alpha^i \to \alpha$, it follows that $\lambda_i \to 1$. Since $\mathbf{z} \in X$, the convexity of the feasible sets of the problems LP($\mathbf{y}^i,\alpha^i$) implies that the $\lambda_i \mathbf{d}^i$ form a sequence of feasible solutions for those problems. Therefore,

$$\limsup h_i^{*L} \le \lim_i h^L(\lambda_i \mathbf{d}^i, \mathbf{y}^i) = h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x}). \quad \square$$

The key factor in the validity proof of the algorithm is the guarantee of a minimum improvement ratio in a neighborhood of non-optimal points. In the following, ρ_0 is a parameter in (0,1), $\mathbf{x}^i \in X$, and $\rho^i(\alpha)$ denotes the improvement ratio corresponding to $\mathbf{x}^i$ with trust region parameter α.

Lemma 4.5 If $\mathbf{x}^i \to \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is not a solution of NLP, then there exists an $\bar{\alpha} > 0$ such that $\rho^i(\alpha) \ge \rho_0$ for all $\alpha \in (0,\bar{\alpha})$ and all $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$.

Proof: By using the uniformity of the approximation error and the continuity of $h^L(\mathbf{d}^L(\alpha,\mathbf{x}),\mathbf{x})$, the proof used for Lemma 4.3 may be extended as needed for this result. $\square$

Theorem 4.6 If $\bar{\mathbf{x}}$ is an accumulation point of a sequence generated by Algorithm LTR, then $\bar{\mathbf{x}}$ is an optimal solution of NLP.

Proof:

Assume the result is false, and let $\{\mathbf{x}^i\}$ be a subsequence such that $\mathbf{x}^i \to \bar{\mathbf{x}}$. Using the preceding lemma, we consider those sufficiently large i such that $\rho^i(\alpha) \ge \rho_0$ for all $\alpha \varepsilon (0,\bar{\alpha})$. Moreover, since the initial value of α for each distinct $\mathbf{x}^i$ is at least $\underline{\alpha}$, it is the case that for arbitrarily large i that $\alpha^i \geq \alpha^* := \min\{\frac{\bar{\alpha}}{\beta},\underline{\alpha}\}$ (since the trust region parameter is not reduced below this quantity to achieve the required improvement ratio) and $\rho^i \geq \rho_0$. Letting $\theta_0 := h^L(\mathbf{d}^L(\alpha^*),\bar{\mathbf{x}})$, we then have

$$h^i(\mathbf{d}^{iL}) \leq \rho_0 \cdot h^{iL}(\mathbf{d}^{iL})$$
$$\leq \rho_0 \cdot \theta_0 /2$$

However, for $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$, the relations

$$f(\mathbf{x}^{i+1}) - f(\mathbf{x}^i) \leq h^i(\mathbf{d}^{iL})$$
$$\leq \rho_0 \cdot \theta_0 /2$$

contradict $f(\mathbf{x}^i) \to f(\bar{\mathbf{x}})$. □

4.3 A PIECEWISE-LINEAR TRUST REGION ALGORITHM

Another type of trust region method for NLP and will be discussed in this section is a piecewise-linear approach, which is developed by Meyer[Meyer 85]. The piecewise-linear trust region algorithm (PLTR) utilizes a separable convex, piecewise-linear approximation to f (note that f itself is not assumed to be separable). The principal advantage is that it is generally more accurate (particularly on the translated axes, where it even enjoys an accuracy advantage over quadratic approximations) than linear approximation, and yet still yields subproblems that may be solved by linear programming.

We now define a separable approximation of h by first considering the single-variable functions obtained by fixing (at $\mathbf{0}$) all variables but one (in this definition, $\mathbf{e}^j$ is the j^{th} unit vector):

$$s_j(d_j) := h(d_j \mathbf{e}^j) \quad (j = 1, \ldots, n).$$

Not that s_j thus corresponds to the restriction of h to the d_j-axis, and the convexity of

f implies the convexity of s_j. A separable approximation of h is then given by summing these single-variable functions:

$$s(\mathbf{d}) := \sum_{j=1}^{n} s_j(d_j) .$$

For computational purposes, we carry out one final approximation step by first defining $\tilde{s}_j(d_j)$ ($j = 1, \cdots ,n$) to be the convex piecewise-linear function obtained by linearly interpolating between the values of s_j at a set of points (to be defined below) on the d_j-axis, and then letting

$$\tilde{h}(\mathbf{d}) := \sum_{j=1}^{n} \tilde{s}_j(d_j) .$$

Note that $\tilde{h}$ is a separable, convex, piecewise-linear function with $\tilde{h}(\mathbf{0}) = 0$, and that $\tilde{h}$ may be thought of as an approximation of the separable function s. Also, we restrict the grid points, on which the piecewise-linear function $\tilde{s}_j$ is defined, to include the point 0 as well as additional grid points spaces not more than α apart.

At each iteration of the algorithm to be defined below, we solve a piecewise-linear approximating problem PL(α,**x**):

$$\min_{d} \tilde{h}(\mathbf{d})$$

$$s.t.\ \ \mathbf{A}\mathbf{d} \le \bar{\mathbf{b}} ,\quad ||\mathbf{d}|| \le \alpha ,$$

We now state the piecewise-linear trust region algorithm:

PLTR Algorithm

step 0:

Select positive α^0, and $\underline{\alpha}$, scalars $0 < \rho_0 < \rho_1 < \rho_2 < 1$ and $\beta > 1$.

Set $i = 0$.

step 1:

Find $\mathbf{x}^0$ satisfying $A\mathbf{x} \le \mathbf{b}$.

step 2:

Solve $PL(\alpha^i, \mathbf{x}^i)$, obtaining an optimal solution $\mathbf{d}^{iPL}$.

step 3:

Compute the actual change $h^i := h(\mathbf{d}^{iPL})$ and predicted change $\tilde{h}^i := \tilde{h}(\mathbf{d}^{iPL})$.

If $h^{iL} = 0$, stop.

Otherwise compute the ratio of actual to predicted change

$$\rho^i := \rho(\alpha^i, \mathbf{x}^i) = \frac{h^i}{\tilde{h}^i}.$$

step 4:

If $\rho^i < \rho_0$ or $\dfrac{h^i}{h^{iL}} < \rho_0$, then $\alpha^i \leftarrow \dfrac{\alpha^i}{\beta}$, goto step 2;

otherwise take $\mathbf{x}^{i+1} \leftarrow \mathbf{x}^i$ and update α^{i+1} by

$$\alpha^{i+1} = \begin{cases} \max\left\{ \dfrac{\alpha^i}{\beta}, \underline{\alpha} \right\} & \text{if } \rho^i \le \rho_1; \\ \max\left\{ \beta\alpha^i, \underline{\alpha} \right\} & \text{if } \rho^i \ge \rho_2; \\ \max\left\{ \alpha^i, \underline{\alpha} \right\} & \text{otherwise.} \end{cases}$$

$i \leftarrow i+1$ and goto step 2.

To prove the convergence theorem for PLTR, we need to have more discussion than that in the preceding section.

Lemma 4.7 Let the grid size for $\tilde{h}(\mathbf{d})$ be no greater than α and let $\|\mathbf{d}\| \le \alpha$. Then the following *error bounds* hold.

(a) $h(\mathbf{d}) - h^L(\mathbf{d}) = o(\|\mathbf{d}\|) = o(\alpha)$

(b) $s(\mathbf{d}) - h^L(\mathbf{d}) = o(\alpha)$

(c) $s(\mathbf{d}) - \tilde{h}(\mathbf{d}) = O(\alpha^2)$

(d) $h^L(\mathbf{d}) - \tilde{h}(\mathbf{d}) = o(\alpha)$

(e) $h(\mathbf{d}) - \tilde{h}(\mathbf{d}) = o(\alpha)$

(f) $h^L(\mathbf{d}^L(\alpha)) - \tilde{h}(\mathbf{d}^{PL}(\alpha)) = o(\alpha)$

Proof: (a) Follows from differentiability of f.

(b) Follows from $\nabla s(\mathbf{0}) = \nabla h(\mathbf{0}) = \nabla h^L(\mathbf{0})$.

(c) This is a property of piecewise-linear approximations of separable functions; see [Feijoo 85].

(d) Follows from (b) and (c).

(e) Follows from (a) and (d).

(f) By part (d) and the definitions of the terms,

$$\tilde{h}(\mathbf{d}^{PL}(\alpha)) \le \tilde{h}(\mathbf{d}^L(\alpha)) \le h^L(\mathbf{d}^L(\alpha)) + o(\alpha).$$

However, since $h^L(\mathbf{d}) \le \tilde{h}(\mathbf{d})$, it follows that $h^L(\mathbf{d}^L(\alpha)) \le h^{PL}(\mathbf{d}^{PL}(\alpha))$, and the combined inequalities $h^L(\mathbf{d}^L(\alpha)) \le h^{PL}(\mathbf{d}^{PL}(\alpha)) \le h^L(\mathbf{d}^L(\alpha)) + o(\alpha)$ yield (f). □

The following lemma establishes some useful convexity properties of the optimal value functions.

Lemma 4.8 For $\lambda \in [0,1]$, $\tilde{h}(\mathbf{d}^{PL}(\lambda\alpha)) \le \tilde{h}(\lambda\mathbf{d}^{PL}(\alpha)) \le \lambda\tilde{h}(\mathbf{d}^{PL}(\alpha))$.

Proof: The left inequality for $\tilde{h}$ follows from the feasibility of $\lambda\mathbf{d}^{PL}(\alpha)$ for the trust region corresponding to $\lambda\alpha$. The right inequality follows from $\mathbf{d}^{PL}(0) = \mathbf{0}$ and the convexity of $\tilde{h}$. □

We now show that an improved value for the approximation $\tilde{h}$ may be obtained by solving PL($\bar{\mathbf{x}}$,α).

Lemma 4.9 If $\mathbf{x}$ is not an optimal solution of (NLP), then for all α sufficiently small, $\tilde{h}(\mathbf{d}^{PL}(\alpha),\mathbf{x}) < 0$.

Proof: Suppose there exists a positive sequence $\alpha_i \to 0$ such that $\tilde{h}(\mathbf{d}^{PL}(\alpha_i)) = 0$ for all i. Since $\tilde{h}(\mathbf{d}^{PL}(\alpha)) = h^L(\mathbf{d}^L(\alpha)) + o(\alpha)$ by Lemma 4.7(f), setting $\alpha = \alpha_i$, dividing through by α_i, and taking limits, yields $\dfrac{h^L(\mathbf{d}^L(\alpha_i))}{\alpha_i} \to 0$. However, since $\bar{\mathbf{x}}$ is non-optimal, for any fixed $\alpha \geq \max_i \alpha_i$ we have by Lemma 4.1 $h^L(\mathbf{d}^L(\alpha)) < 0$ and, by Lemma 4.8, $h^L(\mathbf{d}^L(\dfrac{\alpha_i}{\alpha}\cdot\alpha)) \leq \dfrac{\alpha_i}{\alpha} h^L(\mathbf{d}^L(\alpha))$. Dividing this inequality by α_i yields $\dfrac{h^L(\mathbf{d}^L(\alpha_i))}{\alpha_i} < \dfrac{h^L(\mathbf{d}^L(\alpha))}{\alpha}$, contradicting the convergence of the left-hand-side terms to 0. □

By using convexity properties and error bounds, improvement in the approximating function $\tilde{h}$ may be related to improvement in the objective h.

Lemma 4.10 If for some $\tilde{\alpha}$, $\tilde{h}(\mathbf{d}^{PL}(\tilde{\alpha})) < 0$, then for any $\rho_0 < 1$, the inequality $\rho(\lambda\tilde{\alpha}) \geq \rho_0$ holds for all sufficiently small $\lambda > 0$.

Proof:

$$\begin{aligned}\rho(\lambda\tilde{\alpha}) &= \frac{h(\mathbf{d}^{PL}(\lambda\tilde{\alpha}))}{\tilde{h}(\mathbf{d}^{PL}(\lambda\tilde{\alpha}))} \\ &= \frac{\tilde{h}(\mathbf{d}^{PL}(\lambda\tilde{\alpha})) + o(\lambda\tilde{\alpha})}{\tilde{h}(\mathbf{d}^{PL}(\lambda\tilde{\alpha}))} \\ &= 1 + \frac{o(\lambda\tilde{\alpha})}{\tilde{h}(\mathbf{d}^{PL}(\lambda\tilde{\alpha}))} \\ &= 1 + \frac{o(\lambda\tilde{\alpha})}{\lambda\tilde{h}(\mathbf{d}^{PL}(\tilde{\alpha}))}.\end{aligned}$$

The result follows by noting that the last ratio tends to 0 as $\lambda \to 0$. □

Analogous to Lemma 4.4, we now prove that $\tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x})$ is continuous.

Lemma 4.11 $\tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x})$ is a continuous function of $\mathbf{x}$ and α for $\mathbf{x} \varepsilon X$ and $\alpha > 0$.

Proof: Suppose $(\mathbf{y}^i,\alpha^i) \rightarrow (\mathbf{x},\alpha)$ where $y^i \; \varepsilon X$ and $\alpha^i \geq 0$, and let $\tilde{h}_i^*$ denote the optimal value of the problem corresponding to $(\mathbf{y}^i,\alpha^i)$. By considering an appropriate convergent subsequence of optimal solutions it follows that

$$\liminf \tilde{h}_i^* \geq \tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x}).$$

To establish the other required inequality,

$$\limsup \tilde{h}_i^* \leq \tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x}),$$

we construct a sequence of feasible solutions for the sequence of problems $\mathrm{PL}(\mathbf{y}^i,\alpha^i)$ as follows: let $\mathbf{z} := \mathbf{x} + \mathbf{d}^{PL}(\alpha,\mathbf{x})$, where $\mathbf{d}^{PL}(\alpha,\mathbf{x})$ is any optimal solution of $\mathrm{PL}(\mathbf{x},\alpha)$, (we assume $\mathbf{d}^{PL}(\alpha,\mathbf{x}) \neq \mathbf{0}$, since the result is trivial if $\tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x}) = 0$) and define λ_i and $\mathbf{d}^i$ such that $\mathbf{y}^i + \mathbf{d}^i = \mathbf{z}$ and such that λ_i is the largest scalar in $[0,1]$ such that $\|\lambda_i \mathbf{d}^i\| \leq \alpha^i$. Since $\mathbf{d}^i \rightarrow \mathbf{d}^{PL}(\alpha,\mathbf{x})$ and $\alpha^i \rightarrow \alpha$, it follows that $\lambda_i \rightarrow 1$. Since $\mathbf{z} \varepsilon X$, the convexity of the feasible sets of the problems $\mathrm{PL}(\mathbf{y}^i,\alpha^i)$ implies that the $\lambda_i \mathbf{d}^i$ form a sequence of feasible solutions for those problems. Therefore,

$$\limsup \tilde{h}_i^* \leq \lim_i \tilde{h}(\lambda_i \mathbf{d}^i,\mathbf{y}^i) = \tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x}). \quad \square$$

Let $\rho^i(\alpha)$ denote the improvement ratio corresponding to $\mathbf{x}^i$ with trust region parameter α and $\rho_0 \,\varepsilon\, (0,1)$. Analogous to Lemma 4.5, we have the following lemma.

Lemma 4.12 If $\mathbf{x}^i \rightarrow \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is not a solution of NLP, then there exists an $\bar{\alpha} > 0$ such that $\rho^i(\alpha) \geq \rho_0$ for all $\alpha \,\varepsilon\, (0,\bar{\alpha})$ and all $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$.

Proof: By using the uniformity of the approximation error and the continuity of $\tilde{h}(\mathbf{d}^{PL}(\alpha,\mathbf{x}),\mathbf{x})$, the proof used for Lemma 4.10 may be extended as needed for this result. $\square$

The main convergence theorem of the algorithm now follows in a straightforward manner.

Theorem 4.13 If $\bar{\mathbf{x}}$ is an accumulation point of a sequence generated by Algorithm PLTR, then $\bar{\mathbf{x}}$ is an optimal solution of NLP.

Proof:

Assume the result is false, and let $\{\mathbf{x}^i\}$ be a subsequence such that $\mathbf{x}^i \rightarrow \bar{\mathbf{x}}$. Using the preceding lemma, we consider those sufficiently large i such that $\rho(\mathbf{d}^{iPL}(\alpha),\mathbf{x}^i) \geq \rho_0$ for all $\alpha \, \varepsilon \, (0,\bar{\alpha})$. Moreover, since the initial value of α for each distinct $\mathbf{x}^i$ is at least $\underline{\alpha}$, it is the case that for arbitrarily large i that $\alpha^i \geq \alpha^* := \min\{\frac{\bar{\alpha}}{\beta},\underline{\alpha}\}$ (since the trust region parameter is not reduced below this quantity to achieve the required improvement ratio) and $\rho^i \geq \rho_0$. Letting $\theta_0 := h^L(\mathbf{d}^L(\alpha^*),\bar{\mathbf{x}})$, we then have

$$\begin{aligned} h^i(\mathbf{d}^{iPL}) &\leq \rho_0 \cdot \tilde{h}^i(\mathbf{d}^{iPL}) \\ &\leq \rho_0 \cdot \rho_0 \cdot h^{iL}(\mathbf{d}^{iL}) \\ &\leq \rho_0^2 \cdot \theta_0 /2 \end{aligned}$$

However, for $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$, the relations

$$\begin{aligned} f(\mathbf{x}^{i+1}) - f(\mathbf{x}^i) &\leq h^i(\mathbf{d}^{iPL}) \\ &\leq \rho_0^2 \cdot \theta_0 /2 \end{aligned}$$

contradict $f(\mathbf{x}^i) \rightarrow f(\bar{\mathbf{x}})$. □

For computational efficiency, the algorithm PLTR may be modified by bypassing the initial ratio condition in step 2 whenever $\tilde{h}^i(\mathbf{d}^{iPL}) < -\tau$, where τ is a positive tolerance.

CHAPTER 5

A SCALED PIECEWISE-LINEAR TRUST REGION ALGORITHM

We have discussed two types of trust region methods for general nonlinear program with linear constraints in the preceding chapter. In this chapter we will specify a piecewise-linear trust region method, called SPLT, for nonlinear multicommodity problem (MCP). SPLT differs from PLTR in the two aspects:

(1) The piecewise-linear functions used in SPLT algorithm are based on the scaled separable approximations for the objective function discussed in Chapter 3.

(2) A simple inexact line-search is used at each iteration in SPLT if the improvement ratio check is not satisfied.

Before coming to the algorithm itself, we will give an example in the first section.

5.1 A SKETCH OF THE SPLT ALGORITHM

The example shown below, with three commodities, three nodes, and six arcs, will demonstrate the characteristics of SPLT.

(1) Original Problem :

Consider an MCP as follows:

$$\begin{aligned}\min\ & f_1(x_{11}+x_{21}+x_{31}) + f_2(x_{12}+x_{22}+x_{32}) \\ & + f_3(x_{13}+x_{23}+x_{33}) + f_4(x_{14}+x_{24}+x_{34}) \\ & + f_5(x_{15}+x_{25}+x_{35}) + f_6(x_{16}+x_{26}+x_{36})\end{aligned}$$

$$\begin{array}{ccccl} \mathbf{Ax}_1 & & & = & \mathbf{b}_1 \\ & \mathbf{Ax}_2 & & = & \mathbf{b}_2 \\ & & \mathbf{Ax}_3 & = & \mathbf{b}_3 \end{array}$$

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \geq \mathbf{0}\ .$$

(2) Separable Approximation :

For simplicity, assume 0 is a feasible solution (otherwise shifted functions are used). We can then approximate the objective function to get a separable program:(Scaled separable approximation $\tilde{f}_j$ are used in this step.)

$$\begin{aligned}\min\ & \tilde{f}_1(x_{11})+\tilde{f}_1(x_{21})+\tilde{f}_1(x_{31}) + \tilde{f}_2(x_{12})+\tilde{f}_2(x_{22})+\tilde{f}_2(x_{32}) \\ & + \tilde{f}_3(x_{13})+\tilde{f}_3(x_{23})+\tilde{f}_3(x_{33}) + \tilde{f}_4(x_{14})+\tilde{f}_4(x_{24})+\tilde{f}_4(x_{34}) \\ & + \tilde{f}_5(x_{15})+\tilde{f}_5(x_{25})+\tilde{f}_5(x_{35}) + \tilde{f}_6(x_{16})+\tilde{f}_6(x_{26})+\tilde{f}_6(x_{36})\end{aligned}$$

$$\begin{array}{ccccl} \mathbf{Ax}_1 & & & = & \mathbf{b}_1 \\ & \mathbf{Ax}_2 & & = & \mathbf{b}_2 \\ & & \mathbf{Ax}_3 & = & \mathbf{b}_3 \end{array}$$

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \geq \mathbf{0}\ .$$

(3) Decomposition :

By rearranging the separable terms in commodity order, the problem can thus be decomposed into three subproblems. At this step, the problem can be in turn solved in parallel.

$$\min \tilde{f}_1(x_{11})+\tilde{f}_2(x_{12})+\tilde{f}_3(x_{13})+\tilde{f}_4(x_{14})+\tilde{f}_5(x_{15})+\tilde{f}_6(x_{16})$$
$$\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1 \,,\ \mathbf{x}_1 \geq \mathbf{0}$$

$$\min \tilde{f}_1(x_{21})+\tilde{f}_2(x_{22})+\tilde{f}_3(x_{23})+\tilde{f}_4(x_{24})+\tilde{f}_5(x_{25})+\tilde{f}_6(x_{26})$$
$$\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2 \,,\ \mathbf{x}_2 \geq \mathbf{0}$$

$$\min \tilde{f}_1(x_{31})+\tilde{f}_2(x_{32})+\tilde{f}_3(x_{33})+\tilde{f}_4(x_{34})+\tilde{f}_5(x_{35})+\tilde{f}_6(x_{36})$$
$$\mathbf{A}\mathbf{x}_3 = \mathbf{b}_3 \,,\ \mathbf{x}_3 \geq \mathbf{0}\,.$$

(4) Trust Region :

Impose a trust region with trust region size α_q for each commodity q on the above problem.($\bar{\mathbf{x}}$ denotes the current solution and $||.||$ denotes the infinity norm.)

$$\min \tilde{f}_1(x_{11})+\tilde{f}_2(x_{12})+\tilde{f}_3(x_{13})+\tilde{f}_4(x_{14})+\tilde{f}_5(x_{15})+\tilde{f}_6(x_{16})$$
$$\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1 \,,\ \mathbf{x}_1 \geq \mathbf{0} \,,\ ||\mathbf{x}_1 - \bar{\mathbf{x}}_1|| \leq \alpha_1$$

$$\min \tilde{f}_1(x_{21})+\tilde{f}_2(x_{22})+\tilde{f}_3(x_{23})+\tilde{f}_4(x_{24})+\tilde{f}_5(x_{25})+\tilde{f}_6(x_{26})$$
$$\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2 \,,\ \mathbf{x}_2 \geq \mathbf{0} \,,\ ||\mathbf{x}_2 - \bar{\mathbf{x}}_2|| \leq \alpha_2$$

$$\min \tilde{f}_1(x_{31})+\tilde{f}_2(x_{32})+\tilde{f}_3(x_{33})+\tilde{f}_4(x_{34})+\tilde{f}_5(x_{35})+\tilde{f}_6(x_{36})$$
$$\mathbf{A}\mathbf{x}_3 = \mathbf{b}_3 \,,\ \mathbf{x}_3 \geq \mathbf{0} \,,\ ||\mathbf{x}_3 - \bar{\mathbf{x}}_3|| \leq \alpha_3\,.$$

(5) Piecewise-Linear Approximation :

For each single variable x_{qj}, the corresponding separable function $\tilde{f}_j$ will be approximated by a piecewise-linear function $\hat{f}_{qj}$ (see Fig. 5.1).

$$\min\ \hat{f}_{11}(x_{11})+\hat{f}_{12}(x_{12})+\hat{f}_{13}(x_{13})+\hat{f}_{14}(x_{14})+\hat{f}_{15}(x_{15})+\hat{f}_{16}(x_{16})$$
$$\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1\ ,\ \ \mathbf{x}_1 \geq \mathbf{0}\ ,\ \ \|\mathbf{x}_1-\bar{\mathbf{x}}_1\| \leq \alpha_1$$

$$\min\ \hat{f}_{21}(x_{21})+\hat{f}_{22}(x_{22})+\hat{f}_{23}(x_{23})+\hat{f}_{24}(x_{24})+\hat{f}_{25}(x_{25})+\hat{f}_{26}(x_{26})$$
$$\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2\ ,\ \ \mathbf{x}_2 \geq \mathbf{0}\ ,\ \ \|\mathbf{x}_2-\bar{\mathbf{x}}_2\| \leq \alpha_2$$

$$\min\ \hat{f}_{31}(x_{31})+\hat{f}_{32}(x_{32})+\hat{f}_{33}(x_{33})+\hat{f}_{34}(x_{34})+\hat{f}_{35}(x_{35})+\hat{f}_{36}(x_{36})$$
$$\mathbf{A}\mathbf{x}_3 = \mathbf{b}_3\ ,\ \ \mathbf{x}_3 \geq \mathbf{0}\ ,\ \ \|\mathbf{x}_3-\bar{\mathbf{x}}_3\| \leq \alpha_3\ .$$

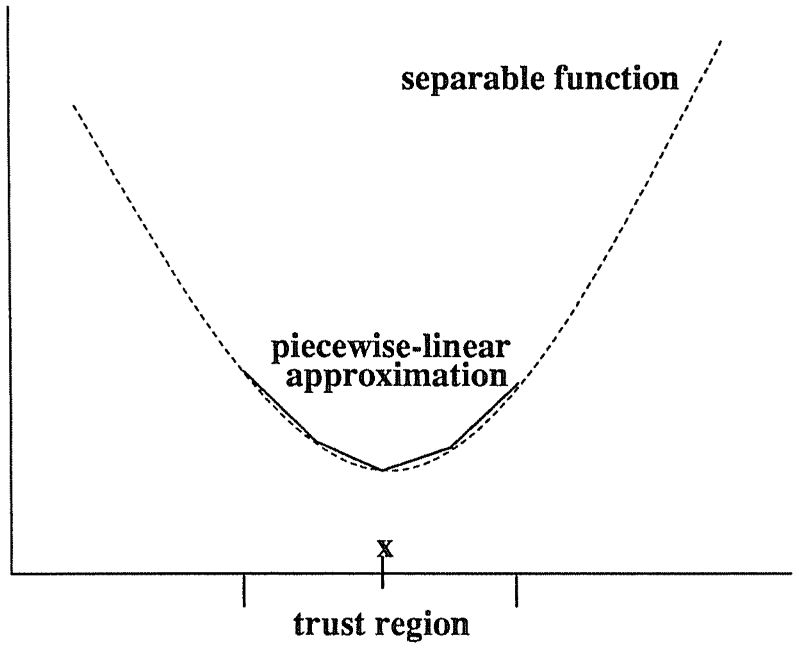


Figure 5.1 The Piecewise-Linear Approximation with 4 Segments for a Separable Function in the Trust Region

(6) Ratio Check :

Let **x** be the current point and **y** be the solution in (5). Denote

$$f^{PL}(\mathbf{x}) := \sum_q \sum_j \hat{f}_{qj}(\mathbf{x}_{qj}).$$

Calculate the improvement ratio $\rho := \dfrac{f(\mathbf{x}) - f(\mathbf{y})}{f^{PL}(\mathbf{x}) - f^{PL}(\mathbf{y})}$.

(1) $\rho \geq 0.3$: accept **y**,

(2) $\rho < 0.3$: get **z** by doing line-search, such as the golden-search or the bisection search, until a good ratio is obtained(that is $\dfrac{f(\mathbf{x}) - f(\mathbf{z})}{f^{PL}(\mathbf{x}) - f^{PL}(\mathbf{y})} \geq 0.3$); or choose **x** if the ratio cannot exceed 0.3 for three evaluations in the line-search.

The choice of the trust region for the next iteration is decided as follows(see Fig. 5.2).

(case 1) $\rho \geq 0.6$: keep the size of trust region unchanged,

(case 2) $0.3 \leq \rho \leq 0.6$: reduce the size of the trust region,

(case 3) $\rho < 0.3$ and cannot get a good ratio in line-search : reduce the size of the trust region,

(case 4) $\rho < 0.3$ and get a good ratio in line-search : reduce the size of the trust region.

In next two sections, we will discuss the SPLT algorithm in more detail.

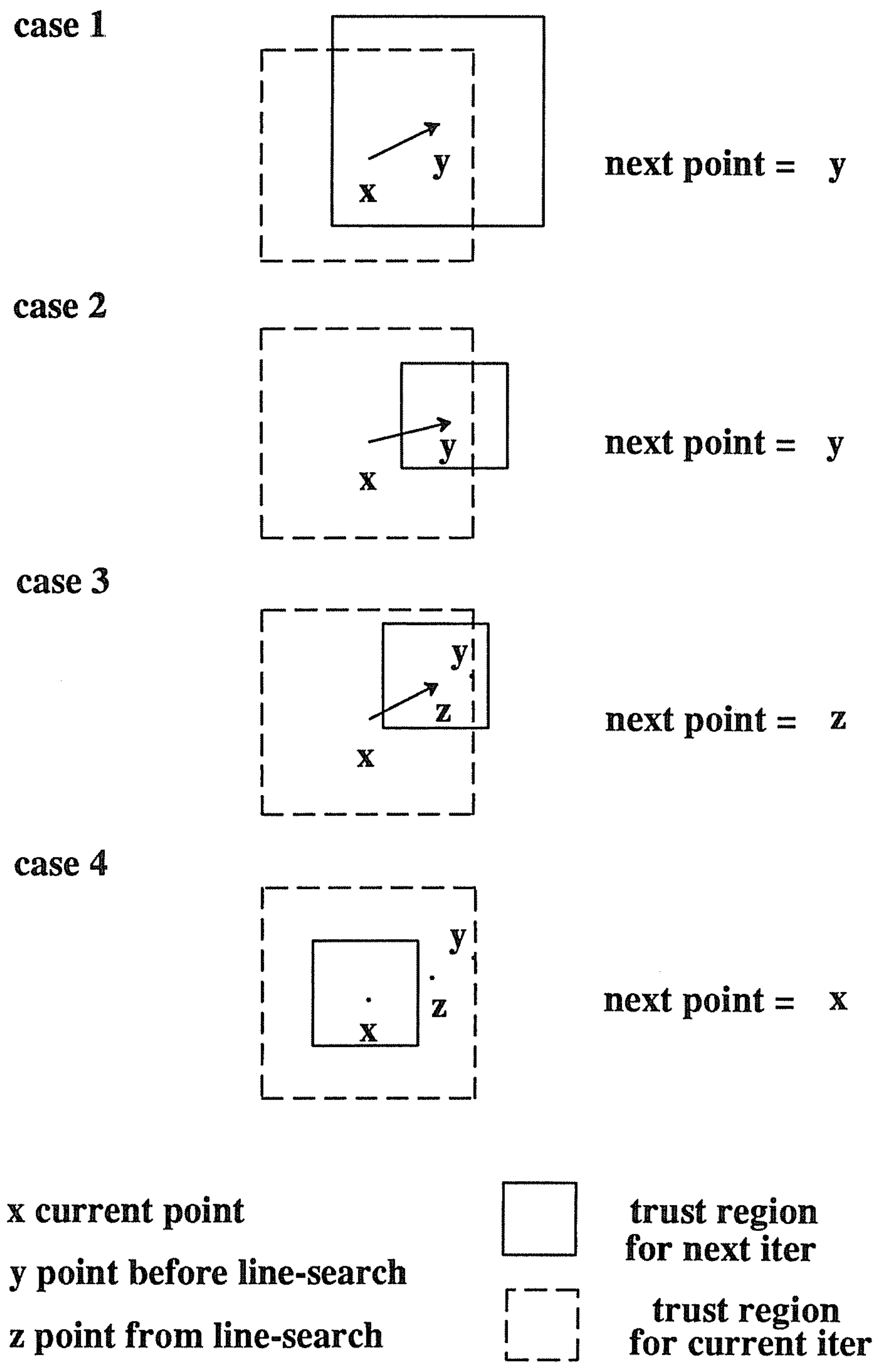


Figure 5.2 The Trust Region Scheme in SPLT

5.2 NOTIONS AND NOTATIONS

For the development of our algorithm, if **x** is a feasible solution of MCP, we define notation as follows:

(1) the trust region :

$\Lambda_{\alpha_q} = \{ \mathbf{d}_q \, \varepsilon R^n \mid -\alpha_q \le d_{qj} \le \alpha_q, j=1,...,n \}$: *trust region* for commodity q

$\Lambda_\alpha = \{ \mathbf{d} \, \varepsilon R^{kn} \mid -\alpha_q \le d_{qj} \le \alpha_q, q=1,...,k, j=1,...,n \}$: *trust region*

$\alpha = (\alpha_1, \cdots, \alpha_k)'$: *trust region* vector

$\underline{\alpha}$: threshold for α

γ_1, γ_2 : reduction factors

(2) linear approximation :

$\mathbf{d}_q^L(\alpha_q)$: a *solution* of $\mathrm{LP}_q(\alpha_q)$ ($\min \nabla h_q(\mathbf{0})\mathbf{d}_q$ *s.t.* $\mathbf{d}_q \, \varepsilon \, \Omega_{\mathbf{x}_q} \cap \Lambda_{\alpha_q}$)

$h^L(\mathbf{d}) := \nabla h(\mathbf{0})\mathbf{d} = \nabla f(\mathbf{x})\mathbf{d}$

$\mathbf{d}^L(\alpha)$: a *solution* of $\mathrm{LP}(\alpha)$ ($\min h^L(\mathbf{d})$ *s.t.* $\mathbf{d} \, \varepsilon \, \Omega_{\mathbf{x}} \cap \Lambda_\alpha$)

(3) separable approximation :

σ : scale factor

$\bar{\sigma}$: upper bound of scale factor

$\mathbf{d}_q^S(\alpha_q, \sigma)$: a *solution* of $\mathrm{SP}_q(\alpha_q, \sigma)$ ($\min h_q^S(\mathbf{d}_q, \sigma)$ *s.t.* $\mathbf{d}_q \, \varepsilon \, \Omega_{\mathbf{x}_q} \cap \Lambda_{\alpha_q}$)

$\mathbf{d}^S(\alpha, \sigma)$: a *solution* of $\mathrm{SP}(\alpha, \sigma)$ ($\min h^S(\mathbf{d}, \sigma)$ *s.t.* $\mathbf{d} \, \varepsilon \, \Omega_{\mathbf{x}} \cap \Lambda_\alpha$)

(4) piecewise-linear approximation :

δ_q : mesh-size used in PL_q ($\delta_q = \alpha_q / 2^i$ for some integer i)

δ : mesh-size vector used in PL

$\mathbf{d}_q^{PL}(\alpha_q, \sigma, \delta_q)$: a *solution* of $\mathrm{PL}_q(\alpha_q, \sigma, \delta_q)$

($\min h_q^{PL}(\mathbf{d}_q, \sigma, \delta_q)$ *s.t.* $\mathbf{d}_q \, \varepsilon \, \Omega_{\mathbf{x}_q} \cap \Lambda_{\alpha_q}$)

$\mathbf{d}^{PL}(\alpha, \sigma, \delta)$: a *solution* of $\mathrm{PL}(\alpha, \sigma, \delta)$

($\min h^{PL}(\mathbf{d}, \sigma, \delta)$ *s.t.* $\mathbf{d} \, \varepsilon \, \Omega_{\mathbf{x}} \cap \Lambda_\alpha$)

(5) improvement ratios :

$\eta(\alpha,\sigma,\delta) := \dfrac{h^{PL}(\mathbf{d}^{PL}(\alpha,\sigma,\delta),\sigma)}{h^{S}(\mathbf{d}^{S}(\alpha,\sigma),\sigma)}$: ratio of piecewise-linear and separable optima

η_0 : threshold of η-ratio ($0 < \eta_0 < 1$)

$\rho(\mathbf{d},\sigma) := \dfrac{h(\mathbf{d})}{h^{PL}(\mathbf{d},\sigma)}$: ratio of h and h^{PL} at $\mathbf{d}$

$\rho_0, \rho_1, \rho_2, \rho_3$: thresholds of ρ-ratio ($0 < \rho_0 < \rho_1 < 1 < \rho_2 < \rho_3$)

(6) the line-search :

L : maximum number of function evaluations per line-search

$\mathbf{d}^{LS}(l)$: lth point in line-search

Where not ambiguous, some arguments of L, S, PL, $\mathbf{d}_q^L$, $\mathbf{d}_q^S$, $\mathbf{d}_q^{PL}$, $\mathbf{d}^L$, $\mathbf{d}^S$, $\mathbf{d}^{PL}$, h, h^L, h^S, h^{PL}, η, ρ are omitted; and the current point on which these are based will be added as needed. For simplicity, we add a superscript i on each item of notation to express the association with the ith current solution $\mathbf{x}^i$.

The η-ratio check in (5) is used only for theoretical purposes. In implementation, we can take the threshold η_0 as a very small positive value so that we don't need to calculate the denominator of η if the numerator is greater than a tolerance. Alternatively, the algorithm SPLT may be modified by bypassing the η-ratio check in **step 2** whenever $h(\mathbf{d}^{PL}) < -\tau$, where τ is a positive tolerance. The proof of the convergence theorem is easily changed to take into account this modification.

The algorithm to be described below determines for each distinct $\mathbf{x}^i$ a trust region vector that provides at least a value η_0 for the η-ratio and a value ρ_0 for the ρ-ratio. This is accomplished in two steps: (1) the bound for the η-ratio is attained by refining the mesh-size as needed in the piecewise-linear subprograms, and (2) the bound for the ρ-ratio is attained either by decreasing the trust region vector α as needed, or by doing a line-search. In addition, we also use a suitable scale factor to adjust the objective approximation in order to achieve ρ_0-ratio quickly. The use of the line-search in our algorithm (when the ρ-ratio bound is not initially satisfied) is not shared by the traditional trust region methods.

5.3 THE SPLT ALGORITHM

```
step 0: initialization:
        Select real numbers $\gamma_1, \gamma_2, \underline{\alpha}, \rho_0, \rho_1, \rho_2, \rho_3, \eta_0, \bar{\sigma}$, and integer $L$;
        Select initial values $\alpha^0$ and $\sigma^0$;
        (with restrictions $0 < \gamma_1 < \gamma_2 < 1$, $0 < \underline{\alpha} < \alpha^0$, $0 < \rho_0 < \rho_1 < 1 < \rho_2 < \rho_3$,
        $0 < \eta_0 < 1, 0 < \sigma^0 < \bar{\sigma}, 0 < L$)
        Set $i = 0$;
        step 1: find a feasible solution:
        Find $x^0 \varepsilon \Omega$;
while tolerance of solution not satisfied do
        step 2: solve piecewise-linear programs:
        $i \leftarrow i+1$;
        ( refine mesh until $\eta$-ratio satisfactory )
        for $q = 1, k$
            $\delta_q^i \leftarrow 2\alpha_q^i$;
            ( reduce mesh size by factor of 1/2 )
            do $\delta_q^i \leftarrow \delta_q^i/2$;
            solve $PL_q(\alpha_q^i, \delta_q^i, x^i)$;
            until $h_q^{PL}(d_q^{PL}(\delta_q^i)) \le \eta_0 \cdot h_q^S(d_q^S)$
        endfor;
        **step 3: do ρ-ratio check:**
        $\rho^i \leftarrow \rho(\mathbf{d}^{PL})$
        if $\rho^i \geq \rho_0$ then
            **step 4: no line-search needed:**
            $\mathbf{x}^i \leftarrow \mathbf{x}^i + \mathbf{d}^{PL}$;
            case ( update trust region size and scaling factor )
                $\rho^i \leq \rho_1$ :  $\alpha^i \leftarrow \max\{\gamma_2\alpha^i, \underline{\alpha}\}$, $\sigma^i \leftarrow \min\{2\sigma^i, \bar{\sigma}\}$;
                $\rho_1 < \rho^i \leq \rho_2$ :  $\alpha^i \leftarrow \max\{\gamma_1\alpha^i, \underline{\alpha}\}$, $\sigma^i \leftarrow \sigma^i$;
                $\rho_2 < \rho^i \leq \rho_3$ :  $\alpha^i \leftarrow \alpha^i$, $\sigma^i \leftarrow 0.75\sigma^i$;
                $\rho^i > \rho_3$ :  $\alpha^i \leftarrow \alpha^i$, $\sigma^i \leftarrow 0.5\sigma^i$;
            endcase;
        else
            **step 5: line-search:**
            ( attempt to satisfy ρ-ratio check by line-search )
            for $l = 1, L$
                $\rho^i \leftarrow \dfrac{h(\mathbf{d}^{LS}(l))}{h^{PL}(\mathbf{d}^{PL})}$;
                if $\rho^i \geq \rho_0$ then
                $\mathbf{x}^i \leftarrow \mathbf{x}^i + \mathbf{d}^{LS}(l)$, $\alpha^i \leftarrow \max\{\gamma_2\alpha^i, \underline{\alpha}\}$, $\sigma^i \leftarrow \min\{2\sigma^i, \bar{\sigma}\}$;
                goto **step 2**;
                endif
            endfor;
            ( ρ-ratio unattained; contract trust region )
            $\alpha^i \leftarrow \gamma_2\alpha^i$; $\sigma^i \leftarrow \min\{2\sigma^i, \bar{\sigma}\}$;
        endif;
    **endwhile**
```

Remarks:

1) In **step 1**, the feasible solution can be found by solving a linear network program. The linear objective function in each commodity can be estimated by $f_j'(0)$ for each arc.
2) In the η-ratio check of **step 2**, $h^S(\mathbf{d}^S(\alpha))$ can be replaced by a lower bound, obtained by the primal method or dual method in [Kamesam and Meyer 84], [Kao and Meyer 81] or by $h^L(\mathbf{d}^L(\alpha))$.
3) If needed, the line-search is used in **step 5**. We use golden-section search until the check is satisfied or until a preset number of evaluations is exhausted, whichever comes first.
4) The initial value of the trust region vector for each distinct $\mathbf{x}^i$ is at least $\underline{\alpha}$, since the value of x^i changes only in **step 4** or in **step 5**, where the new value of α is set to at least $\underline{\alpha}$.

5.4 A CONVERGENCE THEOREM FOR SPLT

Before proving the main convergence theorem, we consider two types of optimality conditions for MCP.

Lemma 5.1 $\mathbf{x}$ is an optimal solution of MCP if and only if $\mathbf{d}^L = \mathbf{0}$ solves LP(α,$\mathbf{x}$) for $\alpha > 0$.

Proof: It is same as Lemma 4.1.

Lemma 5.2 For $\sigma \geq 0$, $\mathbf{x}$ is an optimal solution of MCP if and only if $\mathbf{d}^S = \mathbf{0}$ solves SP(α,σ,$\mathbf{x}$) for $\alpha > 0$.

Proof: The result follows from the above lemma because h^S and and h have the same gradient. □

This lemma implies that if $\mathbf{x}$ is not a solution of MCP, then for any $\alpha > \mathbf{0}$ and $\sigma \geq 0$, $h^S(\mathbf{d}^S(\alpha,\sigma),\sigma) < 0$.

By arguments analogous to those used in Lemma 4.7, it may be shown that for any $\sigma \,\varepsilon\, [0,\bar{\sigma}]$, $h^S(\cdot,\sigma) = h^{PL}(\cdot,\sigma) + o(\alpha) = h(\cdot) + o(\alpha)$. By using the first of these properties, we establish the finiteness of the iteration in step 2 of algorithm SPLT.

Lemma 5.3 For $0 < \eta_0 < 1$, $\alpha > \mathbf{0}$, and $0 \leq \sigma$, if $\mathbf{x}$ is not an optimal solution of MCP, then $\eta(\alpha,\sigma,\delta) \geq \eta_0$ with δ obtained from **step 2.**

Proof: Suppressing arguments to simplify notation, we have

$$h^S(\mathbf{d}^S) \leq h^S(\mathbf{d}^{PL}) \leq h^{PL}(\mathbf{d}^{PL}) \leq h^{PL}(\mathbf{d}^S) = h^S(\mathbf{d}^S) + o(\delta).$$

Thus, $h^{PL}(\mathbf{d}^{PL}) = h^S(\mathbf{d}^S) + o(\delta)$, and since $h^S(\mathbf{d}^S) \neq 0$, the η-ratio bound is obtained as $\delta \to 0$. ☐

The following three lemmas may be proved in a manner analogous to the proofs of Lemma 4.10, 4.11, 4.12.

Lemma 5.4 For $0 < \rho_0 < 1$ and $\bar{\sigma} \geq 0$, if $\mathbf{x}$ is not an optimal solution of MCP, then for α small enough, $\rho(\mathbf{d}^{PL},\sigma) \geq \rho_0$.

In order to guarantee that the improvement ratio behaves properly in the neighborhood of any accumulation point $\bar{\mathbf{x}}$ of a sequence $\mathbf{x}^i$, we now observe that $h^S(\mathbf{d}^S(\alpha,\sigma,\mathbf{x}),\sigma,\mathbf{x})$ is continuous for a fixed scale factor σ.

Lemma 5.5 For $\sigma \geq 0$, $h^S(\mathbf{d}^S(\alpha,\sigma,\mathbf{x}),\sigma,\mathbf{x})$ is a continuous function of $\mathbf{x}$ and α for $\mathbf{x} \,\varepsilon\, \Omega$ and $\alpha \geq \mathbf{0}$.

The key factor in the validity proof of the algorithm is the guarantee of a minimum improvement ratio in a neighborhood of non-optimal points. In the following, ρ_0 is a parameter in (0,1), $\mathbf{y}^i \,\varepsilon\, \Omega$, and $\rho(\mathbf{d},\sigma,\mathbf{y}^i)$ is as defined before.

Lemma 5.6 If $\mathbf{y}^i \to \bar{\mathbf{y}}$, where $\bar{\mathbf{y}}$ is not a solution of MCP, then there exists an $\bar{\alpha} > 0$ such that $\rho(\mathbf{d}^{PL},\sigma,\mathbf{y}^i) \geq \rho_0$ for all $\alpha \,\varepsilon\, (0,\bar{\alpha})$, $\bar{\sigma} \geq \sigma \geq 0$, and all $\mathbf{y}^i$ sufficiently close to $\bar{\mathbf{y}}$.

The main convergence theorem of the algorithm now follows in a straightforward manner.

Theorem 5.7 If $\bar{\mathbf{x}}$ is an accumulation point of a sequence generated by Algorithm SPLT, then $\bar{\mathbf{x}}$ is an optimal solution of MCP.

Proof:

Assume the result is false, and let $\{\mathbf{x}^i\}$ be a subsequence such that $\mathbf{x}^i \to \bar{\mathbf{x}}$. Using the preceding lemma, we consider those sufficiently large i such that $\rho(\mathbf{d}^{iPL}(\alpha),\sigma,\mathbf{x}^i) \geq \rho_0$ for all $\alpha \,\varepsilon\, (0,\bar{\alpha})$, $\sigma \geq 0$. Moreover, since the initial value of α for each distinct $\mathbf{x}^i$ is at least $\underline{\alpha}$, it is the case that for arbitrarily large i that $\alpha^i \geq \alpha^* := \min\{\, \gamma_1\bar{\alpha},\underline{\alpha}\}$ (since the trust region vector is not reduced below this quantity to achieve the required improvement ratio) and $\rho^i \geq \rho_0$. Letting $\theta_0 := h^S(\mathbf{d}^S(\alpha^*),\bar{\sigma},\bar{\mathbf{x}})$, we then have for the iterations without line-search,

$$\begin{aligned} h^i(\mathbf{d}^{iPL}) &\leq \rho_0 \cdot h^{iPL}(\mathbf{d}^{iPL}(\alpha^i,\sigma^i),\sigma^i) \quad (\text{ step } 4\,) \\ &\leq \rho_0 \cdot \eta_0 \cdot h^{iS}(\mathbf{d}^{iS}(\alpha^i,\sigma^i),\sigma^i) \quad (\text{ step } 2\,) \\ &\leq \rho_0 \cdot \eta_0 \cdot h^{iS}(\mathbf{d}^{iS}(\alpha^i,\bar{\sigma}),\bar{\sigma}) \quad (\text{ Lemma } 3.2\,) \\ &\leq \rho_0 \cdot \eta_0 \cdot \theta_0 /2 \quad (\text{ Lemma } 5.5\,) \end{aligned}$$

while for the iterations with line-search, the first inequality above holds for $\mathbf{d}^{iLS}$ from step 5. However, for $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$, the relations

$$\begin{aligned} f(\mathbf{x}^{i+1}) - f(\mathbf{x}^i) &\leq h^i(\mathbf{d}^{iLS}) \textit{ or } h^i(\mathbf{d}^{iPL}) \quad (\text{ step 4 or step 5 }) \\ &\leq \rho_0 \cdot \eta_0 \cdot \theta_0 /2 \end{aligned}$$

contradict $f(\mathbf{x}^i) \to f(\bar{\mathbf{x}})$. $\square$

CHAPTER 6

SEQUENTIAL METHODS

The SPLT algorithm proposed in Chapter 5 is to be called the parallel block Jacobi Algorithm(PBJ) hereafter in order to allow us to contrast this algorithm with a block Gauss-Seidel approach. The term "block" comes from the fact that subproblems are solved which involve blocks of variables related to various commodities. In this chapter, we will introduce another sequential algorithm, called the block Gauss-Seidel Algorithm(BGS), in which we solve for each block by using the most recently updated values of the variables. The performance of BGS is compared with those of another two sequential methods: the Frank-Wolfe trust region method and the reduced gradient method.

6.1 A BLOCK GAUSS-SEIDEL ALGORITHM

The idea behind BGS is that in each minor iteration (corresponding to a single-commodity subproblem in PBJ algorithm), we use the most recently updated information from the other $k-1$ commodities rather than updating all blocks in parallel as in the PBJ algorithm. The philosophy of BGS is like that of the Gauss-Seidel algorithm for linear systems. To help discuss the algorithm further, we introduce the following notation:

Notation

$q^+ := q+1$ if $q \neq k$, $q^+ := 1$ if $q = k$ (*next* commodity index);

$J := \{ 0, k, 2k, \cdots \}$;

α_q : size of trust region for index q;

$\delta_q := \alpha_q$ size of segment in PL approximation for index q (two segment approximation);

$\alpha^0 := (\alpha_1, \cdots, \alpha_k)$ initial size vector of trust region;

$0 < \gamma < 1$: reduction factor for trust region;

$0 < \rho_0 < \frac{1}{k}$: threshold for improvement ratio;

Replacement process with index q :

```
    if $\mathbf{d}_q^{PL}$ satisfies ratio check
        $\mathbf{x}_q \leftarrow \mathbf{x}_q + \mathbf{d}_q^{PL}$;
        $\alpha_q \leftarrow \max\{\alpha_q, \underline{\alpha}_q\}$;
    else
        $\alpha_q \leftarrow \gamma \cdot \alpha_q$;
    endif
```

The block Gauss-Seidel algorithm may now be presented.

BGS Algorithm

```
step 1: ( find a feasible solution )
    Find a feasible solution $\mathbf{x}^0$;
    $j = 0$ (minor iteration index),  $q = 0$;
    $\alpha \leftarrow \alpha^0$;

step 2: ( solve linearized problem periodically )
    if $j \, \varepsilon \, J$
        solve LP($\alpha$, $\mathbf{x}^j$);
        $\theta := h^L(\mathbf{d}^L)$;
    endif
    $q \leftarrow q^+, j \leftarrow j+1$;
step 3: ( solve subproblems and do ratio check )
    Solve $\mathrm{PL}_q(\alpha_q, \delta_q, \mathbf{x}^{j-1})$;
        check  if ratio check for $q$  satisfied ( $\frac{h_q(\mathbf{d}_q^{PL})}{\theta} > \rho_0$ );
        do *replacement process* for index $q$;
        $\mathbf{x}^j \leftarrow$ result of replacement process;
        goto step2
```

6.2 A CONVERGENCE THEOREM FOR BGS

The convergence of BGS is guaranteed by the following theorem. At each iteration (minor iteration), the ratio used in the acceptance criteria is the improvement of the original function over to the improvement of the linear approximation function, which is calculated at the beginning of the major iteration. We need the following lemma in the proof of convergence. Note $\rho_0 \; \varepsilon \; (0, \frac{1}{k})$, where k is the number of the commodities.

Lemma 6.1 If $\mathbf{x}^i \rightarrow \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is not a solution of MCP, then there exists a q^* and an $\bar{\alpha} > \mathbf{0}$ such that $\dfrac{h_{q^*}(\mathbf{d}_{q^*}^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} \geq \rho_0$ for all $\alpha \; \varepsilon \; (\mathbf{0}, \bar{\alpha})$, and all $\mathbf{x}^i$ sufficiently close to $\bar{\mathbf{x}}$.

Proof: Otherwise, for every q and for arbitrarily small α, we have

$$\frac{h_q(\mathbf{d}_q^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} < \rho_0,$$

which implies

$$\frac{\sum_{q=1}^{k} h_q(\mathbf{d}_q^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} < k\rho_0.$$

This contradicts Lemma 4.5. □

Now we will prove the convergence theorem for BGS.

Theorem 6.2 Any accumulation point generated by BGS is an optimal solution of MCP.

Proof:

Let $\{\mathbf{x}^j\}$ be the sequence generated by BGS, $\bar{\mathbf{x}}$ be an accumulation point of X := $\{\mathbf{x}^0\,\mathbf{x}^k, \mathbf{x}^{2k}, \cdots\}$. Let $\{\mathbf{x}^{j_i}\}$ be a subsequence of X such that $\mathbf{x}^{j_i} \to \bar{\mathbf{x}}$. Assume $\bar{\mathbf{x}}$ is not a solution. Two cases will be discussed as follows.

Case 1: ($q^* = 1$ satisfies Lemma 6.1.)

We consider those sufficiently large j_i such that $\rho(\mathbf{d}^{j_i PL}, \mathbf{x}^{j_i}) \geq \rho_0$ for all $\alpha \,\varepsilon\, (0,\bar{\alpha})$. Moreover, since the initial value of α for each distinct $\mathbf{x}^{j_i}$ is at least $\underline{\alpha}$, it is the case that for arbitrarily large j_i that $\alpha^{j_i} \geq \alpha^* := \min\{\gamma\cdot\bar{\alpha}, \underline{\alpha}\}$ (since the trust region vector is not reduced below this quantity to achieve the required improvement ratio) and $\rho^{j_i} \geq \rho_0$. Letting $\theta_0 := h^L(\mathbf{d}^L, \alpha^*, \bar{\mathbf{x}})$, we then have

$$\begin{aligned} h^{j_i}(\mathbf{d}^{j_i PL}) &\leq \rho_0 \cdot \theta^{j_i} \quad (\text{ step 3 }) \\ &\leq \rho_0 \cdot \frac{\theta_0}{2}, \quad (\text{ Lemma 4.4 }) \end{aligned}$$

However, for $\mathbf{x}^{j_i}$ sufficiently close to $\bar{\mathbf{x}}$, the relations

$$\begin{aligned} f(\mathbf{x}^{j_i}+1) - f(\mathbf{x}^{j_i}) &\leq f(\mathbf{x}^{j_i+1}) - f(\mathbf{x}^{j_i}) \\ &\leq h^{j_i}(\mathbf{d}^{j_i PL}) \\ &\leq \rho_0 \cdot \frac{\theta_0}{2} \end{aligned}$$

contradict $f(\mathbf{x}^{j_i}) \to f(\bar{\mathbf{x}})$. So $\bar{\mathbf{x}}$ is a solution. Also $\{f(\mathbf{x}^j)\}$ is a decreasing sequence bounded from below. This implies that any accumulation point of $\mathbf{x}^j$ has objective value $f(\bar{\mathbf{x}})$ and thus is a solution of MCP.

Case 2: ($\dfrac{h_1(\mathbf{d}_1^{PL})}{h^L(\mathbf{d}^L)} < \rho_0$ for all $\mathbf{x}^i \in X$ sufficiently close to $\bar{\mathbf{x}}$.)

We consider those sufficiently large j_i such that $\mathbf{x}^{j_i+1} = \mathbf{x}^{j_i}$. By Lemma 6.1 there exists some commodity index q^* for which improvement ratio is attained. We can choose $\mathbf{x}^{j_i'}$ such that

$$\mathbf{x}^{j_i} = \mathbf{x}^{j_i+1} = \cdots = \mathbf{x}^{j_i'} \neq \mathbf{x}^{j_i'+1}.$$

Note that $\mathbf{x}^{j_i'} \to \bar{\mathbf{x}}$ but the improvement ratio at $\mathbf{x}^{j_i'}$ is greater than ρ_0. The rest of the proof is similar to that in case 1. $\square$

6.3 A COMPARISON OF SEQUENTIAL ALGORITHMS

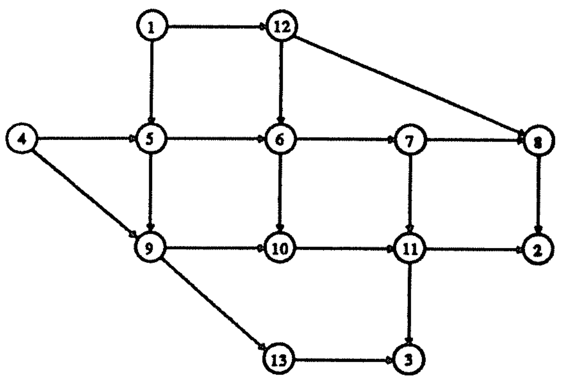
In this section, we will compare the block Gauss-Seidel algorithm with the Frank-Wolfe trust region algorithm [Zhang, et al. 84] and the reduced gradient algorithm (as implemented in MINOS, see [Murtagh and Saunders 78]).

Test Problems

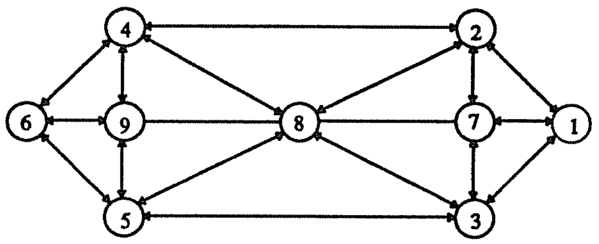
The test data contains five standard traffic assignment problems (see Figure 6.1 for four smaller networks). The sources of these problems are: [Nguyen and Dupuis 84] (Problem A), [Steenbrink 74] (Problem B), [Bertsekas and Gafni 82] (Problem C), [LeBlanc, et al, 75] (Sioux Falls Problem), and [Nguyen and Dupuis 84] (Hull Problem). The Sioux Falls problem and the Hull problem are real traffic assignment problems that model Sioux Falls, South Dakota and Hull, Canada. The dimensions and the objective functions for these five test problems are shown as follows:

Name	Comms	Cnstrs	Vars	Obj Fcn
Problem A	4	52	76	$\sum a_j t_j^2 + b_j t_j$
Problem B	12	108	432	$\sum a_j t_j^2 + b_j t_j$
Problem C	5	125	200	$\sum a_j t_j^3 + b_j t_j^2 + c_j t_j$
Sioux Falls	24	576	1824	$\sum a_j t_j^5 + b_j t_j$
Hull	16	6,768	12,768	$\sum t_j a_j (1 + \frac{\alpha_j}{\beta_j + 1} (\frac{t_j}{b_j})^{\beta_j})$

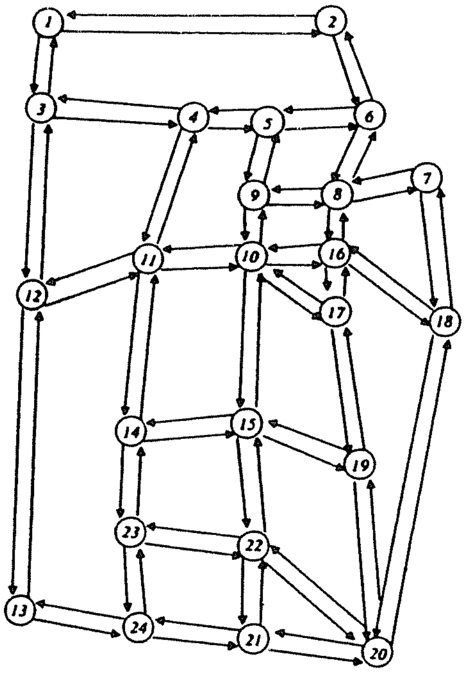
(1) Problem A



(2) Problem B



(4) Sioux Falls Problem



(3) Problem C

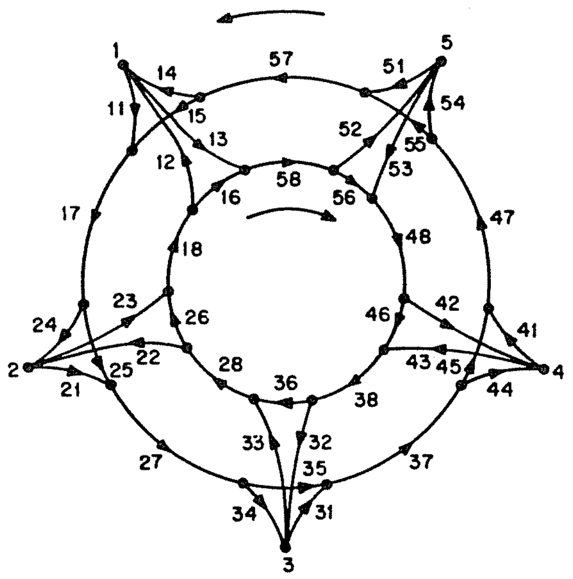


Figure 6.1 Four test problems

Numerical Results

The following table shows the comparison of three sequential algorithms executed on a Microvax II. All the computer codes are written in standard FORTRAN 77 using double precision throughout. The compiler used is Berkeley 4.3 FORTRAN 77 with the optimization compiler option (*–O* flag) set. The timings reported are exclusive of input and output.

Name	Algorithm	Iter	Objective	CPU
Problem A	BGS	14	85028.071	2.9s
	RG	21	85028.071	6.3s
	FWTR	24	85028.071	4s
Problem B	BGS	23	16957.674	18.8s
	RG	106	16957.674	32.9s
	FWTR	57	16957.685	33.1s
Problem C	BGS	23	5924.3427	14.9s
	RG	54	5924.3427	17.5s
	FWTR	35	5934.7617	17.6s
Sioux Falls	BGS	75	721391.91	6m 23s
	RG	733	721391.91	12m 14s
	FWTR	100	730121.42	12m 17s
Hull	BGS	23	31194.605	23m
	RG	3348	31194.645	7h 37m
	FWTR	100	31205.213	1h 21m

It is clear that BGS outperforms both the Frank-Wolfe TR method and the reduced gradient method. The reduced gradient software that we used cannot solve a large-scale problem, like the Hull problem, in a reasonable amount of cpu time, and the Frank-Wolfe method displays slow convergence.

CHAPTER 7

PARALLEL METHODS

In this chapter, we will first introduce the Crystal multicomputer at UW-Madison and then discuss two kinds of SPLT parallel algorithms. The numerical results of the parallel algorithms on Crystal are to be studied as well.

7.1 THE CRYSTAL MULTICOMPUTER

CRYSTAL is a set of 20 VAX-11/750 computers with two megabytes of memory each, connected by a 80 megabit/sec Proteon ProNet token ring(see Fig. 7.1). It can be used simultaneously by multiple research projects by partitioning the available processors according to the requirements of each project. This partitioning is done via the software developed by the Operating System Group at UW-Madison. Once a user has acquired a partition (a subset of processors), he then has exclusive access to the node machines of that partition. CRYSTAL software is written in a local extension to Modula. Researchers can employ the CRYSTAL multicomputer in a number of ways. Projects that need direct control of processor resources can be implemented using a reliable communication service [Cook, et al, 83] that resides on each node processor. Projects that prefer a higher-level interface can be implemented using the Charlotte distributed operating system. The Charlotte kernel provides multiprocessing, inter-process communication, and mechanisms for scheduling, store allocation, and migration. There is also a package called the simple-application package(SAP), which is a set of routines allowing application programmers to use the nugget for communication at a high level. Versions of this package are available for projects using Fortran, Modula, Pascal and C. All the programs discussed below were run on CRYSTAL via SAP and were written in Fortran. The details of the implementation of SAP can be found in [Feijoo 85].

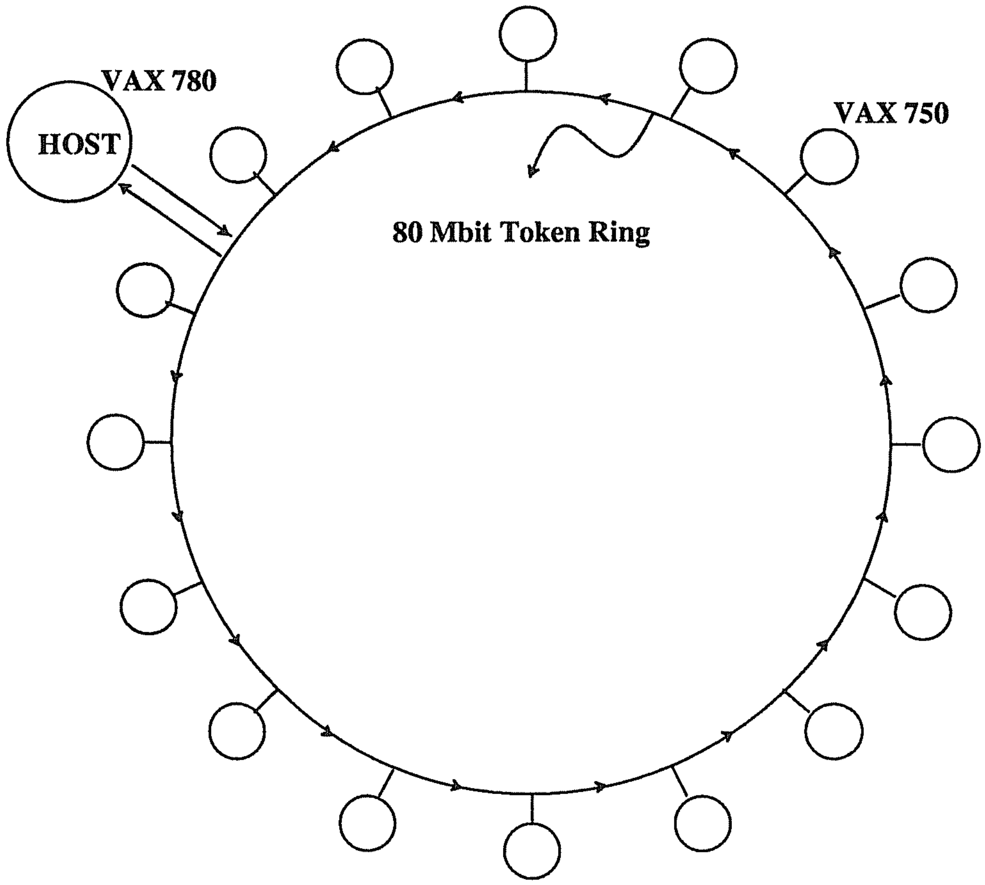


Figure 7.1 CRYSTAL Configuration

7.2 A PARALLEL BLOCK JACOBI ALGORITHM

The block Jacobi algorithm developed in Chapter 6 can be implemented in parallel. Suppose we have gotten a partition with $M+1$ nodes on Crystal, we will assign one node as the master node and the other M nodes as slave nodes. The master node coordinates the parallel solution of the subproblems, and produces new subproblems if the stopping conditions are not satisfied(see Fig. 7.2). Each slave node is assigned $s = \frac{k}{M}$ subproblems(see Fig. 7.3), and thus all the subproblems are solved in parallel.(If k is not divisible by M, some processors are underloaded by one subproblem.) This scheme is called the parallel block Jacobi Algorithm.

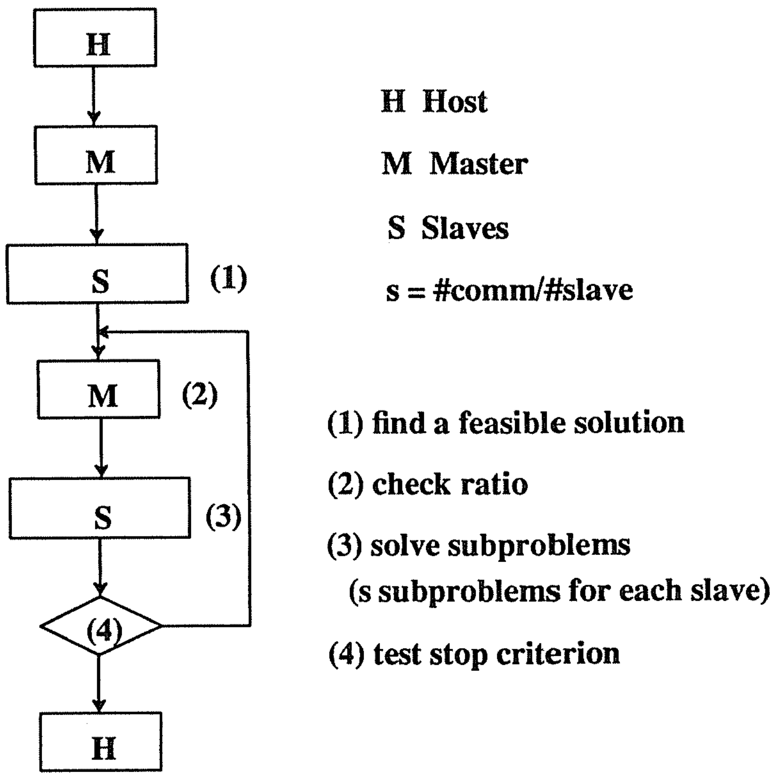
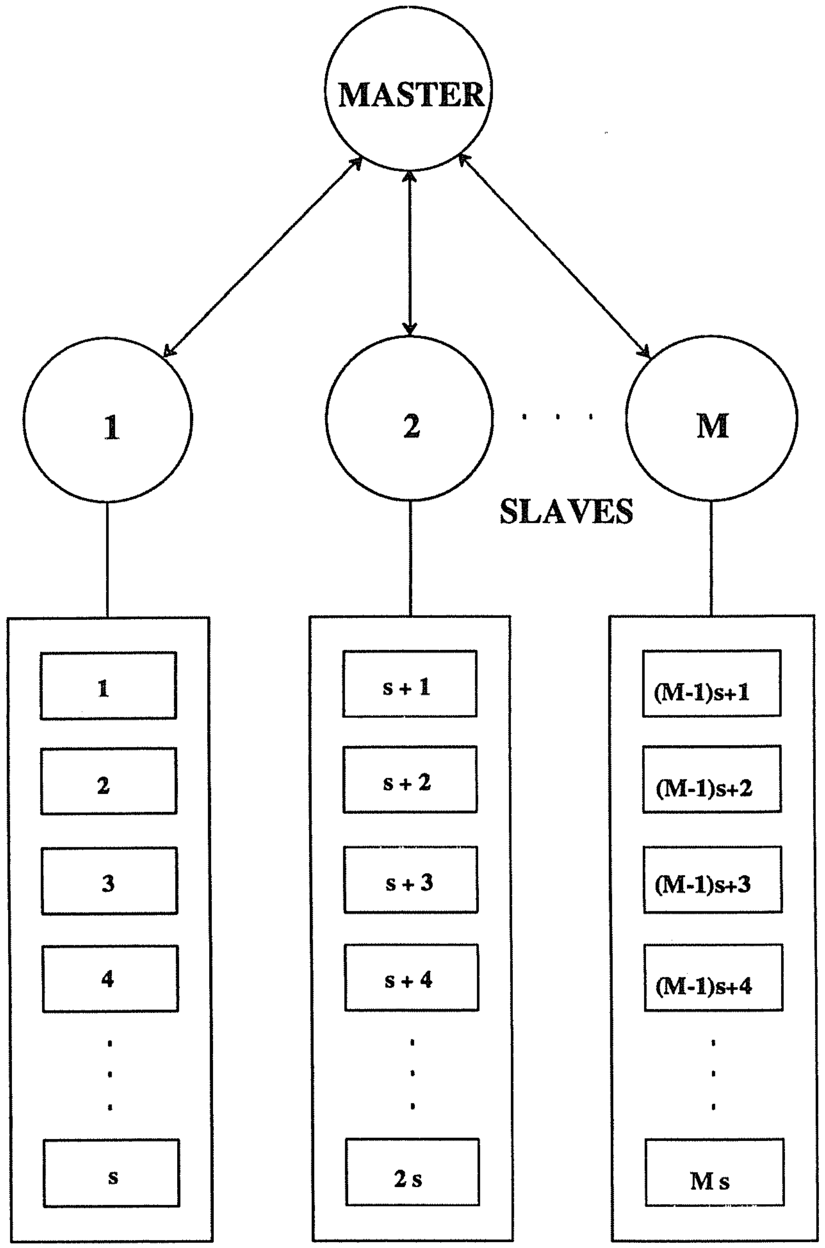


Figure 7.2 A Parallel Block Jacobi Scheme



k subproblems (k = M s)

Figure 7.3 Distribution of Subproblems on Slave Nodes

The following remarks explain certain details in the implementation of PBJ.

(1) initialization:

Choose $\gamma_1 = 0.5$, $\gamma_2 = 0.75$, $\rho_0 = 0.3$, $\rho_1 = 0.8$, $\rho_2 = 1.3$, $\rho_3 = 2$, $L = 3$, and $\sigma_0 = 1$.

(2) feasible solution:

The initial iterate $\mathbf{x}^0$ is taken to be the element of Ω corresponding to the solution of the linear network problem with costs given by $f_j{}'(0)$.

(3) network subroutine:

A modified version of RNET [Grigoriadis and Hsu 79] is used to solve piecewise-linear subproblems.

(4) line-search:

Usually the optimal solution from the trust region is satisfactory, and a line-search is unnecessary. When a line-search is called for, the golden section search is used because of its simplicity. Typically, only one function evaluation is needed in this case to achieve the required ρ-ratio.

(5) stopping criterion:

We calculate a lower bound for MCP every few iterations. Given a current feasible solution $\mathbf{x}^i$, a lower bound is obtained by computing the optimal objective value of the linearized problem :

$$\min f(\mathbf{x}^i) + \nabla f(\mathbf{x}^i)(\mathbf{x}-\mathbf{x}^i) \quad s.t. \ \mathbf{x} \in \Omega$$

(This problem may be decomposed and solved in parallel.) If the gap between the current objective value and the best lower bound is below the preset tolerance, then stop. The test results indicate that the lower bounds obtained from the linearized linear programs are not very tight relative to the upper bounds from the feasible solutions. We also stop the algorithm if the improvements of the objective functions are sufficiently small in three consecutive iterations. The eight-figure agreement in the objective values produced by PBJ and MINOS for most of the test problems suggests that the results are correct to more significant figures than the lower bound would indicate.

Speedup for PBJ

We will use the same test problems as in the preceding chapter. The speedup of $M+1$ nodes is defined as $\frac{cpu \text{ for } 1 \; node}{cpu \text{ for } M+1 \; nodes}$. Table 7.1 and Fig. 7.4 show the cpu time and the speedup for PBJ for five test problems.

Analysis of the Speedup

The causes of the gap between the actual speedup and the ideal speedup are communication time and waiting time. Experience shows that the communication time is too small and can be neglected. As for waiting time, there are three types:

(1) when the master node waits for slave nodes,

(2) when faster slave nodes wait for slower slave nodes, and

(3) when slave nodes wait for the master node.

Figure 7.5 demonstrates these three types of waiting time by dotted lines and demonstrates working time by solid lines.

Problem A	**PBJ (21 Iters)**		
Machines	1	3	5
CPU	24.6s	15.3s	11.0s
Speedup	1	1.6	2.3
Efficiency	1	0.54	0.45

Problem B	**PBJ (37 Iters)**					
Machines	1	3	4	5	7	13
CPU	2m 13s	1m 14s	52.7s	42.8s	33.7s	24.3s
Speedup	1	1.8	2.5	3.1	4.0	5.5
Efficiency	1	0.60	0.63	0.62	0.57	0.42

Problem C	**PBJ (35 Iters)**	
Machines	1	6
CPU	1m 24.7s	24.8s
Speedup	1	3.4
Efficiency	1	0.57

Sioux Falls Problem	**PBJ (100 Iters)**						
Machines	1	3	4	5	7	9	13
CPU	27m 53s	14m 45s	10m 17s	8m 1s	5m 47s	4m 30s	3m 25s
Speedup	1	1.9	2.7	3.5	4.8	6.2	8.2
Efficiency	1	0.63	0.68	0.70	0.69	0.69	0.63

Hull Problem	**PBJ (44 Iters)**				
Machines	1	3	5	9	17
CPU	1h 59m	1h 2m	32m 33s	18m 39s	9m 21s
Speedup	1	1.9	3.7	6.4	12.7
Efficiency	1	0.64	0.73	0.71	0.74

Table 7.1 Performance for PBJ

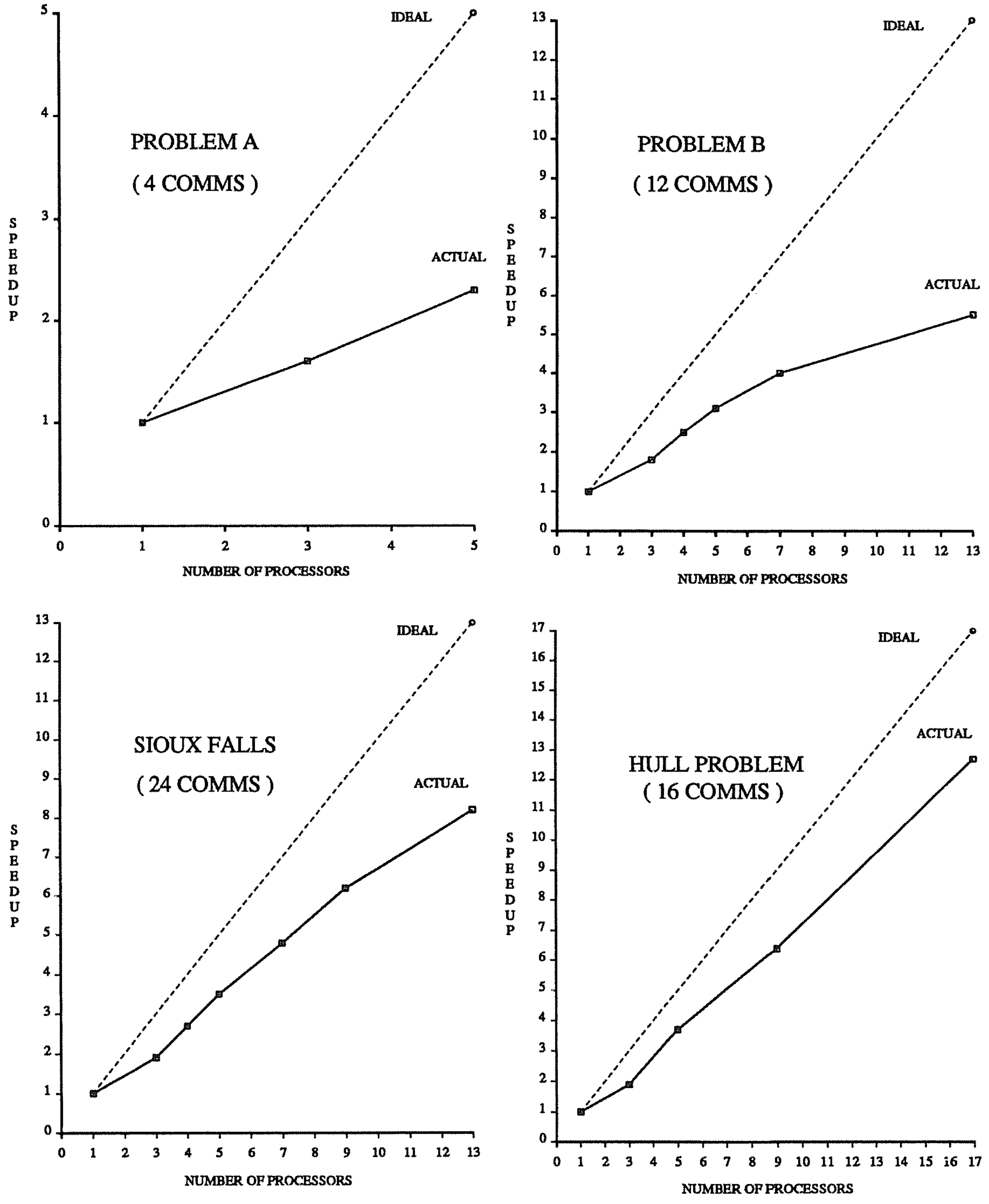


Figure 7.4 Speedup for PBJ

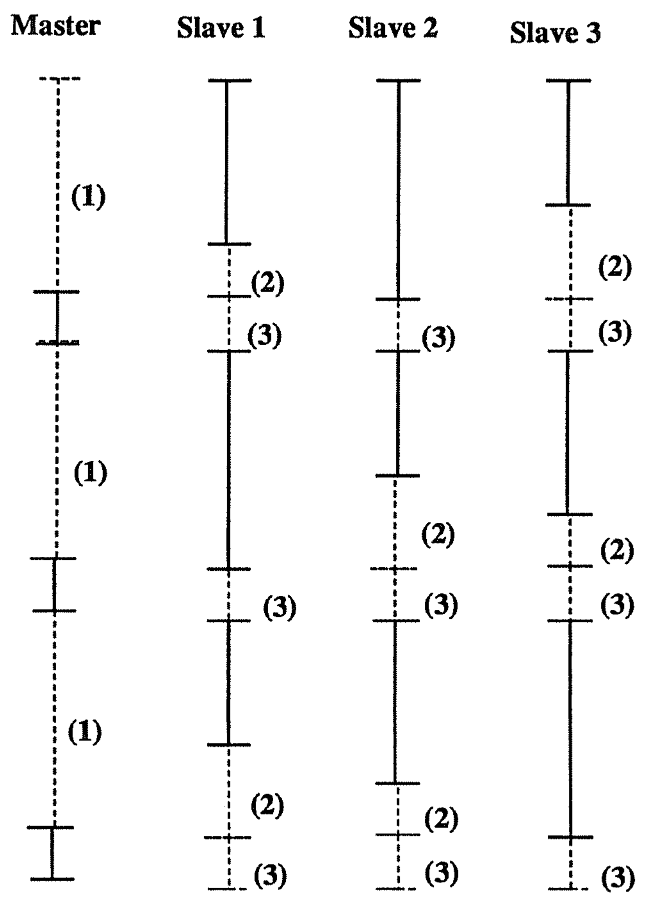


Figure 7.5 Waiting Time for PBJ

Load Balancing

Efficiency may be improved by employing a variety of load balancing tools, including

(1) Reallocate the subproblems to idle processors.

(2) Set a maximal cpu time for each subproblem.

(3) Let the fastest slave node send "stop" message to the other slave nodes.

(4) Let the master and slaves work in parallel.

In the next section, we discuss a strategy that uses the last of these devices.

7.3 A PARALLEL BLOCK G-S ALGORITHM

To increase the efficiency of this decomposition approach, we develop in this section a parallel processor utilization technique that combines concepts related to block Gauss-Seidel and block Jacobi procedures. In this approach, each commodity flow vector corresponds to a block of variables, and a number of blocks equal to the number of processors are updated in parallel (via an optimization procedure applied to each block independently) at each major iteration. The acceptability of the updated values for such a group of blocks (in terms of the required amount of improvement of the original objective function) is then checked by a coordination processor while the remaining processors work on the next group of blocks, which assume the un-updated values for the previous group. However, by the time this next group has been optimized, the coordination check for the previous group has been accomplished, and the updated information (if it has met the acceptability criteria) may then be utilized in setting up the initial conditions for the following group. This procedure, which uses efficiently the multiprocessor environment, may be demonstrated to be convergent to the optimal solution of the original problem(see proof below). As expected, it also displays a better convergence rate than the analog of the block Jacobi method previously used. Details of this technique will be described below.

If there are M available slave nodes on the Crystal multicomputer, and k is the number of commodities, we assume $k = Ms$ for simplicity. We can distribute subproblems $1+(p-1)s$, $2+(p-1)s$, $\cdots$, ps to slave node p, for $p = 1, \cdots, M$ (see Fig. 7.2).

To help discuss the algorithm further, we define additional notation as follows.

Notation :

M = # slave machines;

(each machine solves $s = \frac{k}{M}$ subproblems.)

(r th machine solves $(r-1)s+1, (r-1)s+2, .., rs$ th subproblems.)

$\mathbf{y}_t \leftarrow (\mathbf{x}_t, \mathbf{x}_{t+s}, .., \mathbf{x}_{t+(M-1)s})$;

($\mathbf{y}_t$ is *group* of blocks corresponding to group index t.)

$t^- := t-1$ if $t \neq 1$, $t^- := s$ if $t = 1$(*previous* group index);

$t^+ := t+1$ if $t \neq s$, $t^+ := 1$ if $t = s$ (*next* group index);

$J := \{ 0, s+1, 2s+1, \cdots \}$;

j : minor iteration index;

$\mathbf{y}^j$: *jth* iterate;

α_t : size vector of trust region for index t; approximation;

$\delta_t := \alpha_t$ size vector of segment in PL approximation for index t (two-segment approximations);

$\alpha^0 : (\alpha_1, \cdots, \alpha_s)$ initial size vector of trust region;

$0 < \gamma < 1$: reduction factor for trust region;

$0 < \sigma < M$: scalar for PL approximation;

$0 < \rho_0 < \frac{M}{k} (= \frac{1}{s})$: threshold for improvement ratio;

```
Replacement process with index $t$ :
    if $\mathbf{d}_t^{PL}$ satisfies ratio check
        $\mathbf{y}_t \leftarrow \mathbf{y}_t + \mathbf{d}_t^{PL}$; (other groups unchanged)
        $\alpha_t \leftarrow \max\{\alpha_t, \underline{\alpha_t}\}$;
    else
        $\alpha_t \leftarrow \gamma \cdot \alpha_t$;
    endif
```

The parallel block Gauss-Seidel piecewise-linear trust region algorithm may now be presented.

PBGS Algorithm

```
step 1: ( find a feasible solution )
    Find a feasible solution $\mathbf{y}^0$;
    $j = 0$ (minor iteration index),  $t = 0$;
    $\alpha \leftarrow \alpha^0$;

step 2: ( solve linearized problem periodically )
    if $j \in J$
        solve LP($\alpha$, $\mathbf{y}^j$);
        $\theta := h^L(\mathbf{d}^L)$;
    endif
    $t \leftarrow t^+, j \leftarrow j+1$;
step 3: ( solve subproblems and do ratio check )
    Solve PL$_t$($\alpha_t$, $\delta_t$, $\sigma$, $\mathbf{y}^{j-1}$);
        send to Master -  ratio check for $t$;
        check  if ratio check for $t^-$ satisfied ( $\frac{h_{t^-}(\mathbf{d}_{t^-}^{PL})}{\theta} > \rho_0$ );
        ( ratio check unavailable for $j$=1 )
        do *replacement process* for index $t^-$;
        $\mathbf{y}^j \leftarrow$ result of replacement process;
        ( $\mathbf{y}^1 := \mathbf{y}^0$ )
        goto step2
```

Remarks

PBGS is analogous to BGS but different from it in two points:

(1) a group of blocks corresponding to M commodities are solved at each minor iteration, and

(2) the replacement process at the current minor iteration is to update the group of blocks for the previous iteration.

Convergence

The convergence proof of PBGS is analogous to the convergence proof of BGS in section 6.2. At each iteration (minor iteration), the ratio used is the improvement of the original function relative to the improvement of the linear approximation function, which is calculated after first group subproblems are solved. In the following lemma, $\rho_0 \, \varepsilon \, (0, \frac{M}{k})$.

Lemma 7.1 If $\mathbf{y}^i \rightarrow \bar{\mathbf{y}}$, where $\bar{\mathbf{y}}$ is not a solution of MCP, then there exists a t^* and an $\bar{\alpha} > \mathbf{0}$ such that $\dfrac{h_{t^*}(\mathbf{d}_{t^*}^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} \geq \rho_0$ for all $\alpha \, \varepsilon \, (\mathbf{0}, \bar{\alpha})$, and all $\mathbf{y}^i$ sufficiently close to $\bar{\mathbf{y}}$.

Proof: Otherwise, for every t and for arbitrarily small α, we have

$$\frac{h_t(\mathbf{d}_t^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} < \rho_0,$$

which implies

$$\frac{\sum_{t=1}^{s} h_t(\mathbf{d}_t^{PL}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} < s\,\rho_0.$$

This contradicts Lemma 4.5. □

Now we will prove the convergence theorem for PBGS.

Theorem 7.2 Any accumulation point generated by PBGS is an optimal solution of MCP.

Proof:

Let $\{y^j\}$ be the sequence generated by PBGS, $\bar{y}$ be an accumulation point of $Y := \{y^0\, y^{s+1}, y^{2s+1}, \cdots \}$. Let $\{y^{j_i}\}$ be a subsequence of Y such that $y^{j_i} \rightarrow \bar{y}$. Assume $\bar{y}$ is not a solution. Two cases will be discussed as follows.

Case 1: ($t^* = 1$ satisfies Lemma 7.1.)

We consider those sufficiently large j_i such that $\rho(d^{j_i PL}, y^{j_i}) \geq \rho_0$ for all $\alpha \,\varepsilon\, (0,\bar{\alpha})$. Moreover, since the initial value of α for each distinct y^{j_i} is at least $\underline{\alpha}$, it is the case that for arbitrarily large j_i that $\alpha^{j_i} \geq \alpha^* := \min\{\gamma \cdot \bar{\alpha}, \underline{\alpha}\}$ (since the trust region vector is not reduced below this quantity to achieve the required improvement ratio) and $\rho^{j_i} \geq \rho_0$. Letting $\theta_0 := h^L(d^L, \alpha^*, \bar{y})$, we then have

$$h^{j_i}(d^{j_i PL}) \leq \rho_0 \cdot \theta^{j_i} \quad \text{(step 3)}$$
$$\leq \rho_0 \cdot \frac{\theta_0}{2}, \quad \text{(Lemma 4.4)}$$

However, for y^{j_i} sufficiently close to $\bar{y}$, the relations

$$f(y^{j_i}+1) - f(y^{j_i}) \leq f(y^{j_i+1}) - f(y^{j_i})$$
$$\leq h^{j_i}(d^{j_i PL})$$
$$\leq \rho_0 \cdot \frac{\theta_0}{2}$$

contradict $f(y^{j_i}) \rightarrow f(\bar{y})$. So $\bar{y}$ is a solution. Also $\{f(y^j)\}$ is a decreasing sequence bounded from below. This implies that any accumulation point of y^j has objective value $f(\bar{\mathbf{y}})$ and thus is a solution of MCP.

Case 2: ($\dfrac{h_1(\mathbf{d}_1^{PL})}{h^L(\mathbf{d}^L)} < \rho_0$ for all $\mathbf{y}^i \in Y$ sufficiently close to $\bar{\mathbf{y}}$.)

We consider those sufficiently large j_i such that $\mathbf{y}^{j_i+1} = \mathbf{y}^{j_i}$. By Lemma 7.1 there exists some group index t^* for which improvement ratio is attained. We can choose $\mathbf{y}^{j_i'}$ such that

$$\mathbf{y}^{j_i} = \mathbf{y}^{j_i+1} = \cdots = \mathbf{y}^{j_i'} \neq \mathbf{y}^{j_i'+1}.$$

Note that $\mathbf{y}^{j_i'} \to \bar{\mathbf{y}}$ but the improvement ratio at $\mathbf{y}^{j_i'}$ is greater than ρ_0. The rest of the proof is similar to that in case 1. □

7.4 PERFORMANCE FOR PBGS

Test Problems

We used five test problems for PBGS. The Sioux Falls problem and the Hull problem have been described in Chapter 6. The following shows the other three.

Name	Comms	Cnstrs	Vars	Obj Fcn
Gen	24	4,608	12,864	$\sum \frac{1}{2} t_j^2 + t_j$
Mini-Winnipeg	24	28,960	68,064	$\sum t_j a_j (1 + \frac{\alpha_j}{\beta_j + 1} (\frac{t_j}{b_j})^{\beta_j})$
Winnipeg	135	140,400	382,860	$\sum t_j a_j (1 + \frac{\alpha_j}{\beta_j + 1} (\frac{t_j}{b_j})^{\beta_j})$

Gen problem is a generated problem as shown in Fig. 7.6. The nodes on the left-hand-side of the graph are origins and those on the right-hand-side are destinations. We can generate a class of large-scale problems by the integer pairs p and q.

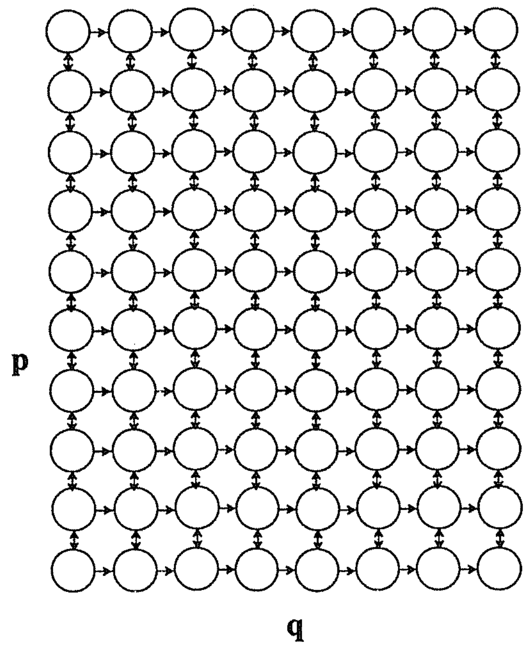


Figure 7.6 A Generated Network

The Winnipeg problem models the traffic in Winnipeg, Canada. The Mini-Winnipeg problem is a smaller version using the first 24 commodities.

Precision for PBGS

Intuitively, if we use fewer nodes for PBGS we may get more rapid convergence because the approximating separable function uses more recent information at each iteration. Figures 7.7-7.10 show the rate of convergence using different numbers of nodes.

Speedup for PBGS

Regardless of the precision of the solution, we fix the number of iterations and define the speedup as before. The speedups for the first four test problems are shown in Fig. 7.11.

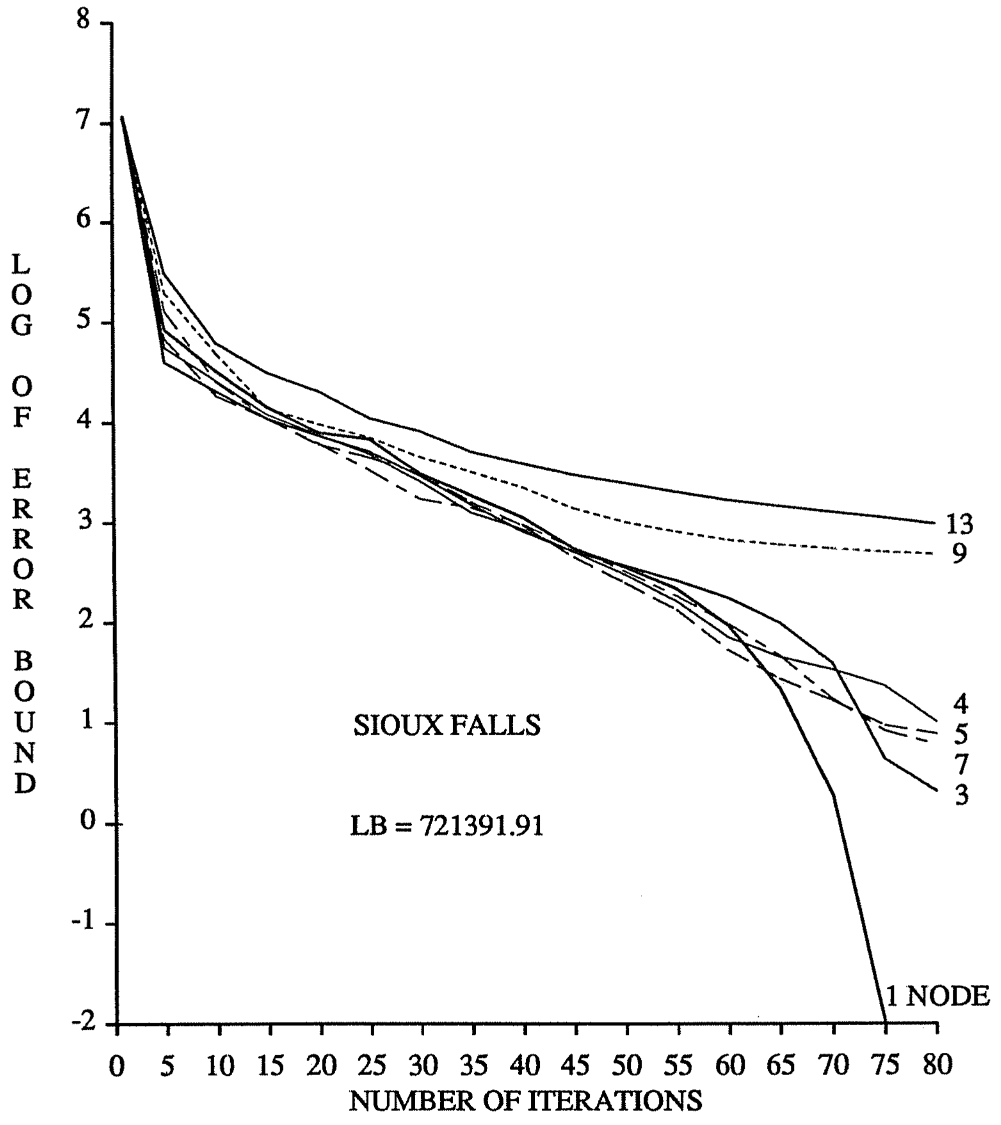


Figure 7.7 Precision for PBGS (SIOUX FALLS)

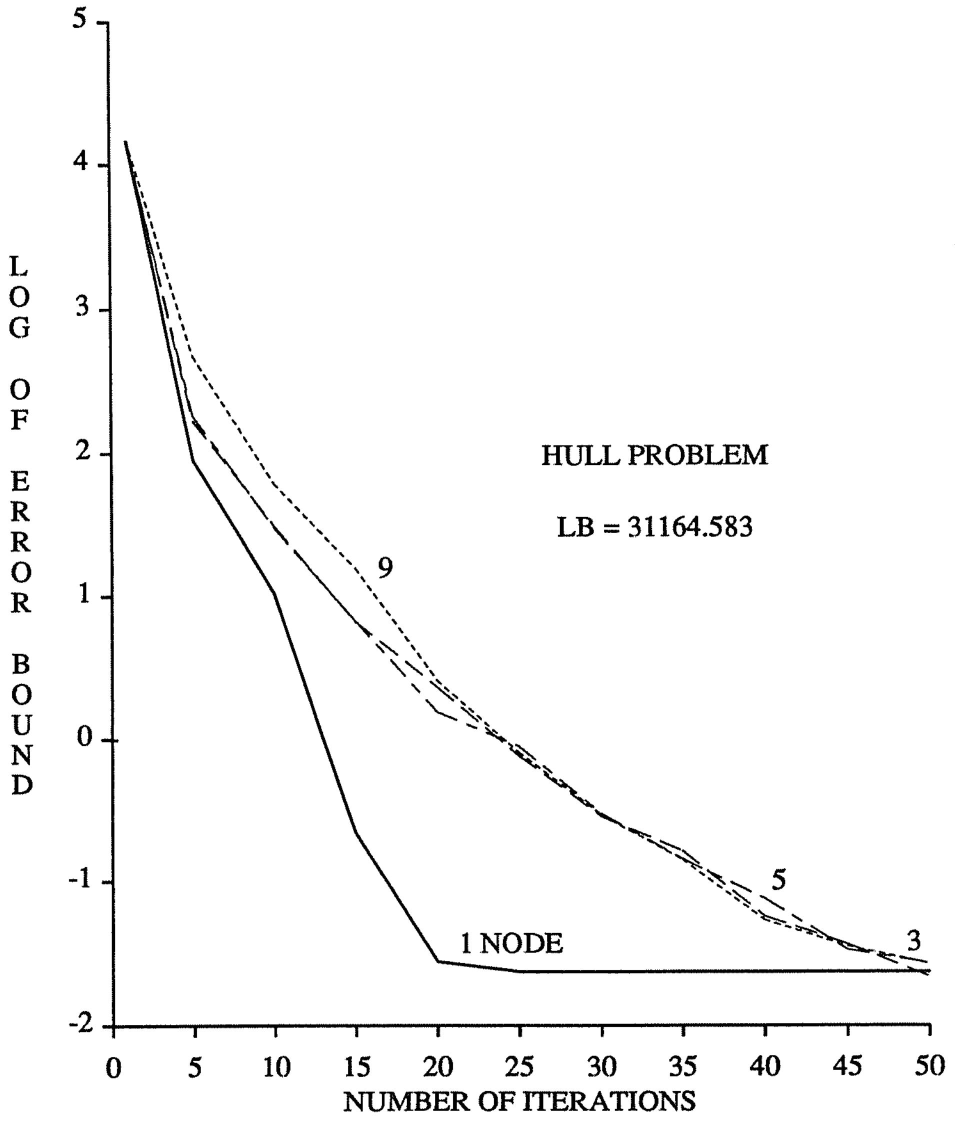


Figure 7.8 Precision for PBGS (HULL)

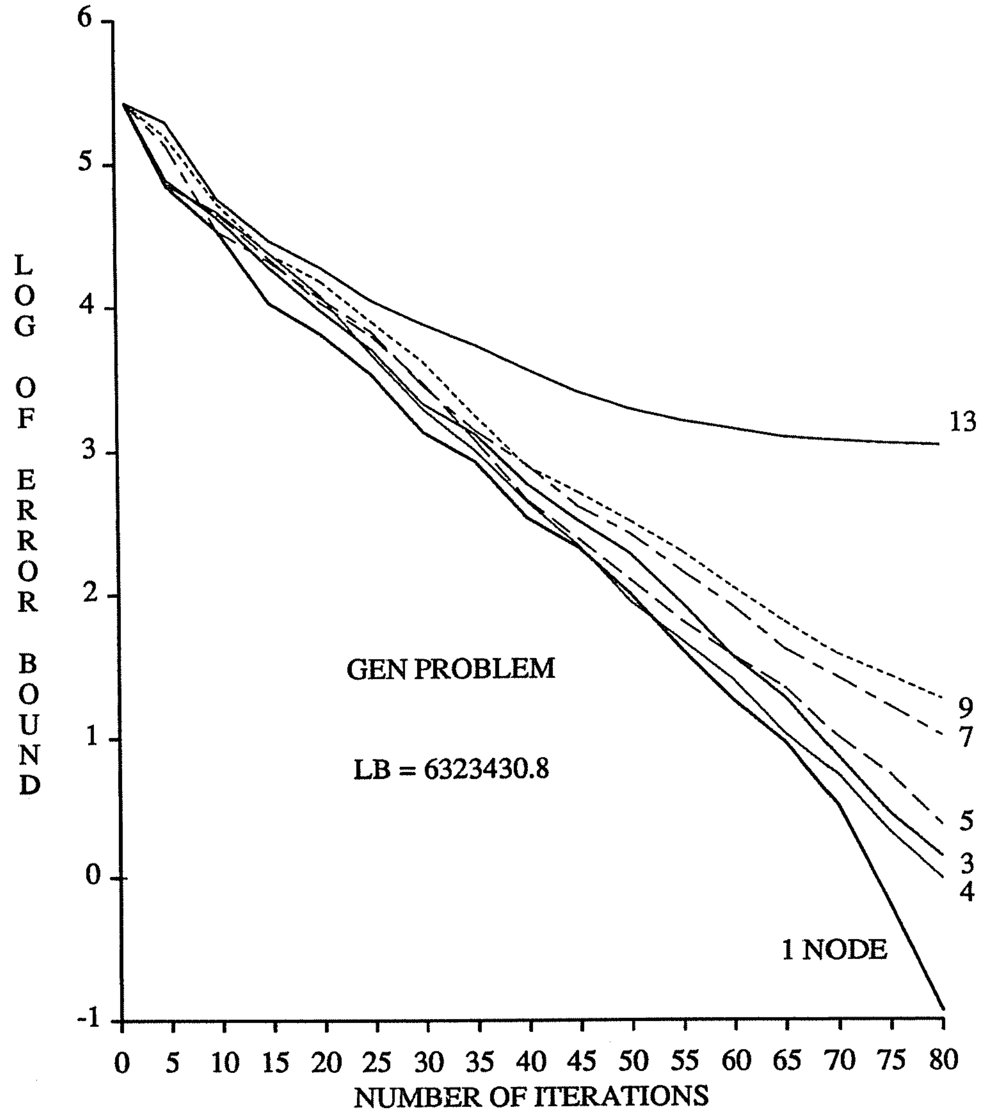


Figure 7.9 Precision for PBGS (GEN)

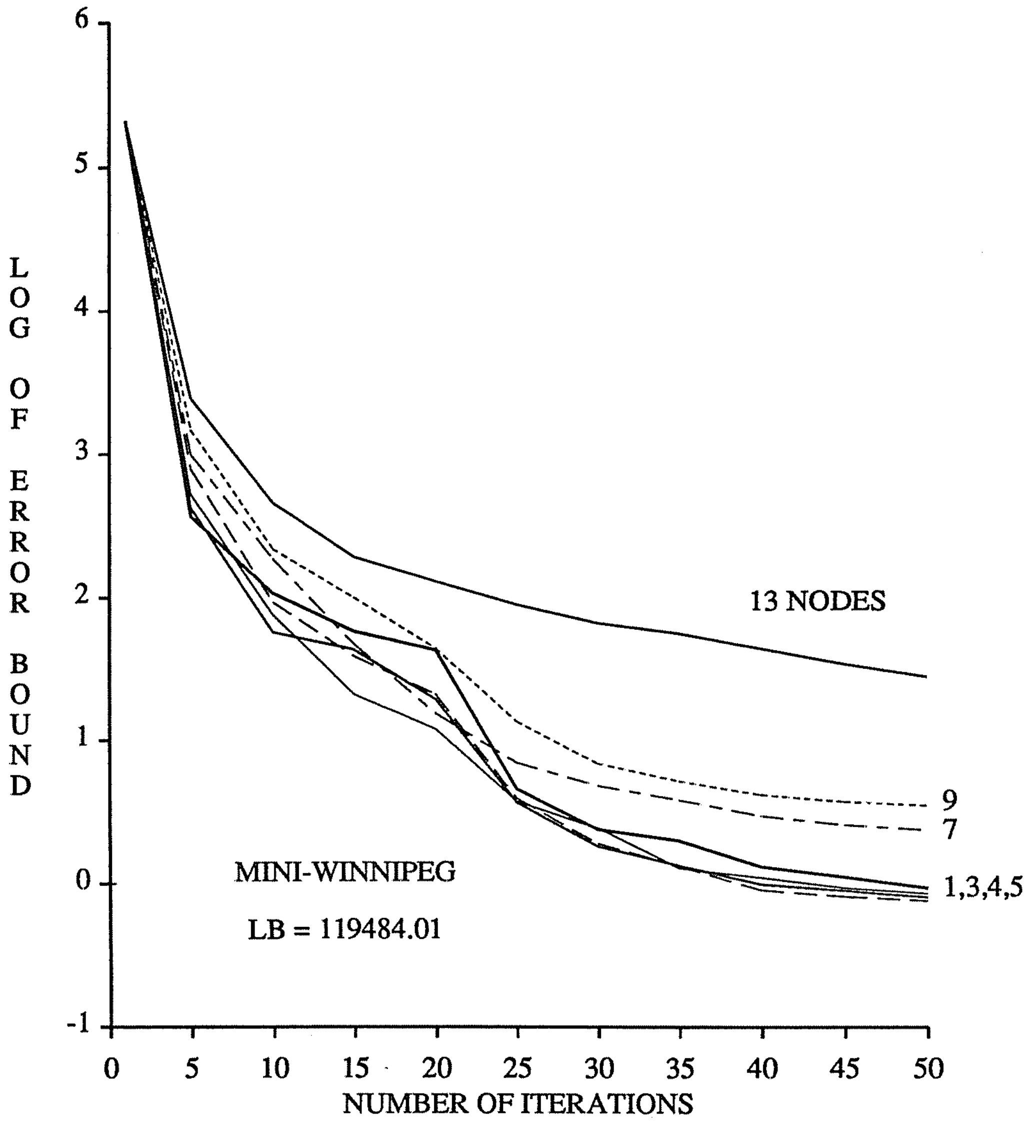


Figure 7.10 Precision for PBGS (MINI-WINNIPEG)

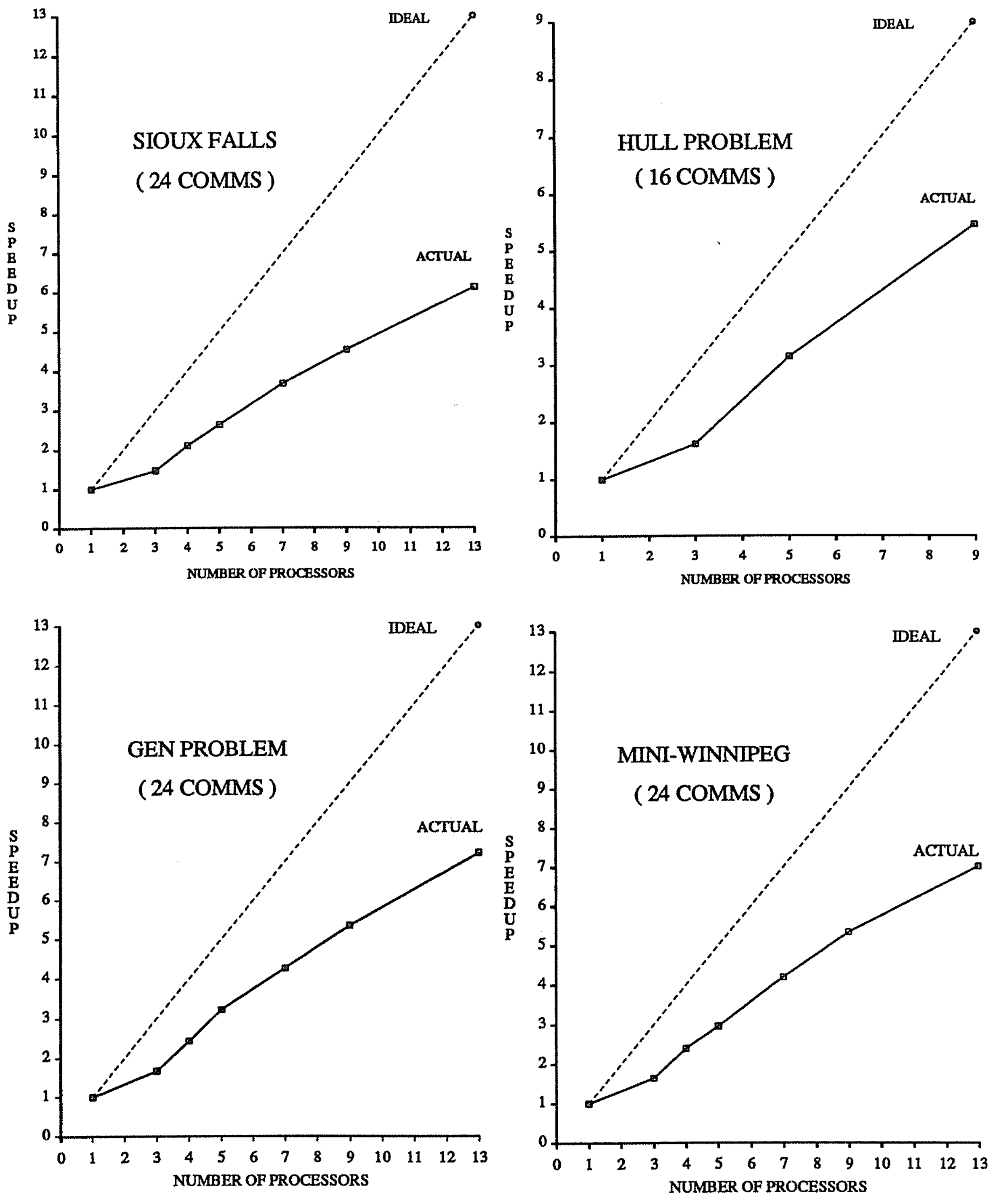


Figure 7.11 Speedup for PBGS

The Winnipeg problem is the biggest test problem, with nearly four hundred thousand variables. The following table shows the solution values obtained by BGS and PBGS after twenty iterations on one node, ten nodes, and sixteen nodes and Fig. 7.12 shows the speedups. A rough lower bound is calculated by linearizing the original function at the solution point obtained from BGS.

# Mach	Iter	Obj	CPU
1	20	623103.62	30h
10	20	623278.96	3h 50m
16	20	623371.54	2h 28m

Best Lower Bound = 622643.35

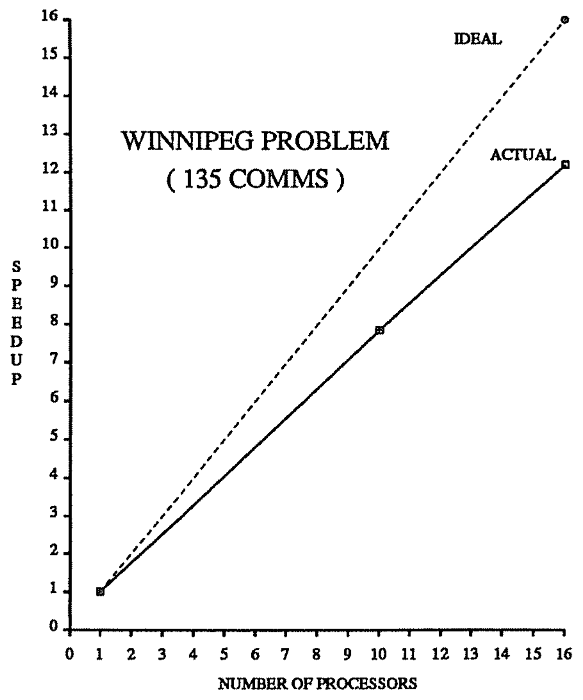


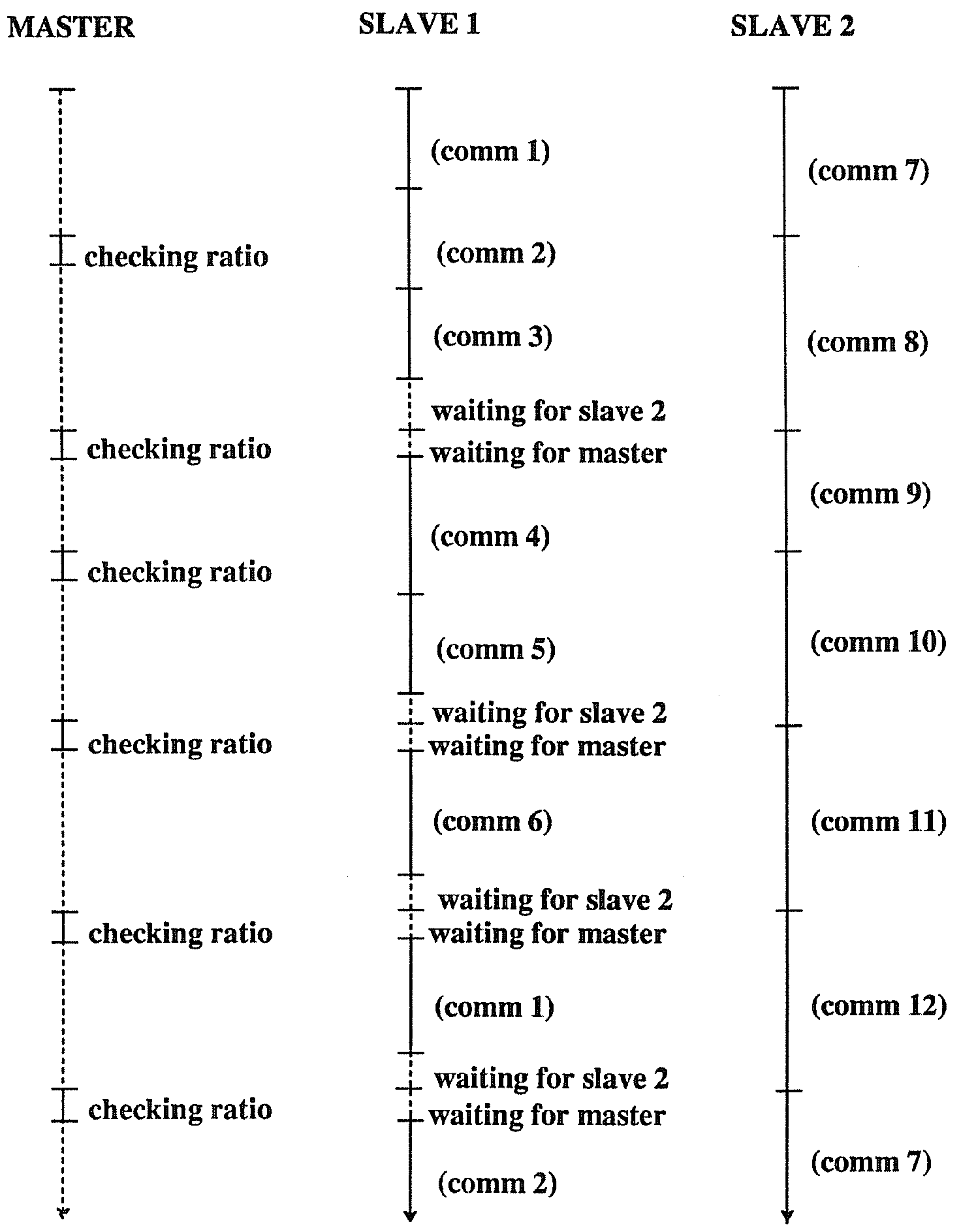
Figure 7.12 Speedup for PBGS (Winnipeg Problem)

Analysis of Speedup

We will count solving a group of subproblems as a minor iteration. The slave nodes need to send a group of blocks of solutions to the master node and the master node needs to send the "acceptance" or "rejection" message back to the slave nodes in each minor iteration. Ignoring communication time, let I_1 be the smallest integer I such that $\sum_{i=1}^{I} cpu_{n_1}(i) < \sum_{i=1}^{I-1} cpu_{n_M}(i)$, where $cpu_{n_j}(i)$ denotes the cpu time for ith minor iteration on node n_j, where n_j denotes the jth fastest slave node.(For simplicity we assume that n_1 and n_M remain the same for all iterations.) The first waiting time for the slaves occurs after n_1 finishes the I_1th minor iteration. Figure 7.13 shows the working time and waiting time for a simple example with one master node and two slave nodes, where where $n_1 = 1$ and $n_2 = 2$. The dotted lines denote the waiting time and the solid lines denote the working time. After the slowest slave node sends solutions to the master node, some of the slower nodes have data available to solve the next subproblems, while some of the faster nodes have to wait for the previous subproblem data. At this point, we can define a delay function d for each node as:

$$d_{n_j} := (\text{ the starting time for } I_1{+}1th \text{ minor iteration on node } n_j \text{ })$$
$$- (\text{ the starting time for } I_1{+}1th \text{ minor iteration on node } n_1 \text{ }).$$

Note that the delay function of the fastest node is zero (i.e. $d_{n_1} = 0$). Let I_2 be the smallest integer I such that $\sum_{i=I_1+1}^{I} cpu_{n_1}(i) < d_{n_M} + \sum_{i=I_1+1}^{I-1} cpu_{n_M}(i)$, The second waiting time occurs after n_1 finishes I_2th minor iteration. This property may be generalized in the obvious manner. After the first wait occurs for n_M, later waits for n_M will occur more often due to the delay for n_M in starting the minor iterations. This is the main factor of the difference between the actual speedup and ideal speedup in Fig. 7.11-12.



(M=2 , k=12)

Figure 7.13 Waiting Time for PBGS

CHAPTER 8

DIRECTIONS FOR FURTHER RESEARCH

In this chapter we discuss directions for further research both in the theoretical and computational areas. Section 8.1 will deal with the communication routing problem, which contains extra coupling constraints in addition to the same block structure as the multicommodity problem. A share-memory multiprocessor--the Sequent Balance is introduced in section 8.2, and the implications of this alternative architecture for the parallel implementation of the algorithm are considered.

8.1 COMMUNICATION ROUTING PROBLEMS

An extension of the traffic assignment problem is optimal routing in packet-switched computer communication networks[Cantor and Gerla 74]. In a packet-switched computer communication network, messages are segmented into packets. The packets are stored in queues at intermediate nodes until communication channels become free. The ARPA Computer Network(see Fig. 8.1) is a packet-switched communication network connecting several computer facilities in the United States. The mathematical model for this type of routing problem is a TAP with objective function:

$$T := \sum_{j=1}^{n} f_j(t_j) = \sum_{j=1}^{n} \frac{1}{\gamma} \cdot \frac{t_j}{p_j - t_j}$$

where

T = total average delay per packet [seconds/packet];

r_{uv} = average packet rate from source u to destination v [packet/second];

$\gamma := \sum_{u=1}^{m} \sum_{v=1}^{m} r_{uv}$ total packet arrival rate from external sources [packet/second];

t_j = total bit rate on channel j [bits/second];

p_j = capacity of channel j [bits/second].

The communication routing problem contains coupling constraints for all arcs due to the capacity restrictions on the arcs. These constraints complicate the process of decomposing the original problem into subproblems. It would be of interest to extend the results of this thesis to allow both implicit(via the objective function) and explicit(via nested decomposition) treatment of additional coupling constraints. Future research may have to find ways to overcome this difficulty.

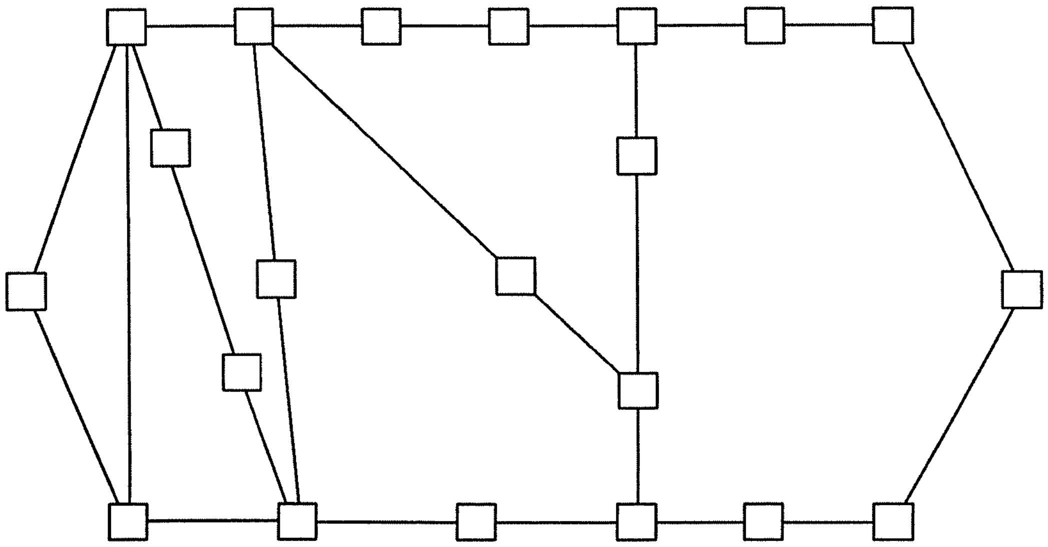


Figure 8.1 A 21-Node ARPA Topology

8.2 PARALLEL ALGORITHMS FOR MULTIPROCESSORS

The Sequent Balance is a multiprocessor, a computer that incorporates multiple identical processors (CPUs) and a single common memory. The CPUs are general purpose, 32-bit microprocessors. The systems are available in two models, the Sequent Balance 8000 and the Sequent Balance 21000. The Sequent Balance 8000 can include from 2 to 12 processors, while the Sequent Balance 21000 can include from 4 to 30 processors. Both models can be configured with 4 to 28 Mbytes of memory and both provide up to 16 Mbytes of virtual address space per process. In addition, each CPU has 8 Kbytes of local RAM and 8 Kbytes of cache RAM, both of which greatly reduce the number of times the processor must access system memory.

The Sequent Balance supports the two basic kinds of parallel programming: *multiprogramming* and *multitasking*. Multiprogramming is an operating system feature that allows a computer to execute multiple unrelated programs concurrently. Multitasking is a programming technique that allows a single application to consist of multiple processes executing concurrently. The following characteristics distinguish the Sequent Balance architecture from the Crystal architecture:

1. *shared memory-* An application can consist of multiple instruction streams, all accessing shared data structures in memory.

2. *common bus-* All processors, memory modules, and I/O controllers plug into a single high-speed bus.

3. *transparency-* Programs written for a single-processor system can run on a Sequent Balance system without modifications for multiprocessing support. Processors can be added or removed without modifying the operating system or user applications.

4. *dynamic load balancing-* Processors automatically schedule themselves to ensure that all processors are kept busy as long as there are executable processes

available. When a processor stops executing, it begins executing the next available process in the system-wide run queue.

5. *hardware support for mutual exclusion-* To support exclusive access to shared data structures, the system includes one or more sets of 16K user-accessible hardware locks.

Both the Parallel Block Jacobi algorithm and the Parallel Block Gauss-Seidel algorithm of Chapter 7 can be implemented as a multitasking program on the Sequent Balance. Furthermore, by the characteristics of the architecture, the Sequent Balance provides the capability of implementing a variety of good parallel algorithms. Future research may generate nice algorithms under this class of multiprocessors.

A parallel scheme using M processors, which does not involve much waiting time, is as follows:

We allocate M commodities to M processors and put the remaining $k-M$ commodities on the run queue. Each processor does the folowing jobs in parallel:(see Fig. 8.2, where an arrow from a processor represents completion of processing of a commodity and an arrow to a processor represents acquisition of another commodity)

(1) solve the subproblem of the allocated commodity,

(2) check ratio,

(3) move the current commodity to the tail of the run queue,

(4) acquire the first commodity on the run queue.

Idle time is only spent waiting until common memory can be accessed.

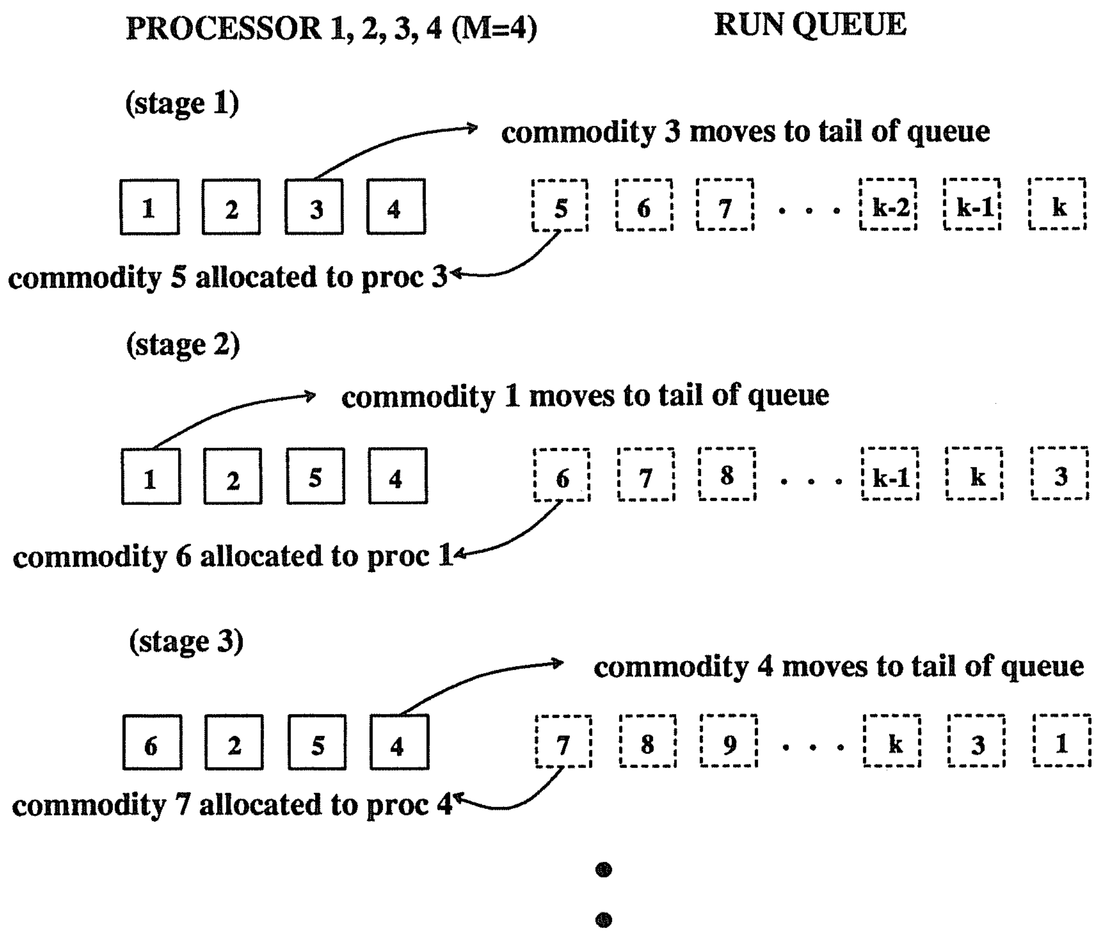

PROCESSOR 1, 2, 3, 4 (M=4)
RUN QUEUE
(stage 1)
commodity 3 moves to tail of queue
1 2 3 4
5 6 7 . . . k-2 k-1 k
commodity 5 allocated to proc 3
(stage 2)
commodity 1 moves to tail of queue
1 2 5 4
6 7 8 . . . k-1 k 3
commodity 6 allocated to proc 1
(stage 3)
commodity 4 moves to tail of queue
6 2 5 4
7 8 9 . . . k 3 1
commodity 7 allocated to proc 4

Figure 8.2 A Parallel Scheme on the Multiprocessor

A similar asynchronous method may be implemented on Crystal with the master doing all of the ratio checks.

BIBLIOGRAPHY

Beck, P., Lasdon, L., and Engquist, M. [1983]: "A reduced gradient algorithm for non-linear network problems", *ACM Transactions on Mathematical Software, Vol. 9, No. 1, 57-70.*

Bertsekas, D. P. and Gafni, E. M. [1982]: "Projection methods for variational inequalities with application to the traffic assignment problem", *Mathematical Programming Study 17, 139-159.*

Cantor, D. G. and Gerla, M. [1974]: "Optimal routing in packet switched computer networks", *IEEE Transactions on Computing C-23, 1062-1068.*

Collins, M., Cooper, L., Helgason, R. V., Kennington,J. L., and LeBlanc, L. J. [1978]: "Solving the pipe network analysis problem using optimization techniques", *Management Science, 24, 747-760.*

Cook, R., Finkel, R., DeWitt, D., Landweber, L., and Virgilio, T. [1983]: "The Crystal Nugget", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #499.

Dafermos, S. C. [1980]: "Traffic equilibrium and variational inequalities", *Transportation Science 14, 42-54.*

Dantzig, G., Harvey, R., Lansdowne, Z., Robinson, D., and Maier, S. [1979]: "Formulating and solving the network design problem by decomposition", *Transportation Res. 13B, 5-17.*

Dantzig, G. B. and Wolfe, P. [1960]: "Decomposition principle for linear programs", *Operations Res. 8, 101-111.*

Dembo, R. S. and Klincewicz, J. G. [1981]: "A scaled reduced gradient algorithm for network flow problems with convex separable costs", *Mathematical Programming Studies, 15, 125-147.*

DeWitt, D., Finkel, R., and Solomon, M. [1984]: "The CRYSTAL multicomputer: design and implementation experience", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #553.

Dijkstra, E. [1959]: "A note on two connexion with graphs", *Numerische Mithematic, 1, 269-171.*

Feijoo, B. [1985]: "Piecewise-linear approximation methods and parallel algorithms in optimization", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #598.

Feijoo, B. and Meyer, R. R. [1984]: "Piecewise-linear approximation methods for non-separable convex optimization", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #521.

Feijoo, B. and Meyer, R. R. [1985]: "Optimization on the Crystal multicomputer", in *Computing 85,* G. Bucci and G. Valle eds., Elsevier Science Publishers.

Fletcher, R. [1981]: *Practical Methods of Optimization,* John Wiley.

Grigoriadis, M. D. and Hsu, T. [1979]: "RNET the Rutgers minimum cost network flow subroutine", *SIGMAP BULLETIN, 17-18.*

Hanscom, M. A., Lafond, L., Lasdon, L. and Pronovost, G. [1980]: "Modeling and resolution of the medium term energy planning problem for a large hydro-electric system", *Management Science, 26, 659-668.*

Johnson, D. B. [1973]: *Algorithms for Shortest Paths,* Dept. of Computer Sciences, Cornell Univ.

Kamesam, P. V. and Meyer, R. R. [1984]: "Multipoint methods for separable nonlinear networks", *Mathematical Programming Study 22, 185-205.*

Kao, C. Y. and Meyer, R. R. [1981]: "Secant approximation methods for convex optimization", *Mathematical Programming Study 14, 143-162.*

Klincewicz, J. G. [1983]: "A Newton method for convex separable network flow problems", *Networks 13, 427-442.*

Lawphongpanich, S. and Hearn, D. W. [1983]: "Restricted simplicial decomposition with application to the traffic assignment problem", University of Florida Department of Industrial and System Engineering Research Rpt. 83-8.

LeBlanc, L. J., Morlok, E. K., and Pierskalla, W. P. [1975]: "An efficient approach to solving the road network equilibrium traffic assignment problem", *Transportation Res. 9, 309-318.*

Leventhal T. L., Nemhauser G. L., and Trotter Jr., L. E. [1973]: "A column generation algorithm for optimal traffic assignment", *Transportation Science 7, 168-176.*

Magnanti, T. L. and Wong, R. T. [1984]: "Network design and transportation planning: models and algorithm", *Transportation Science 18, 1-55.*

McCallum, C. J. [1976]: "A generalized upper bounding approach to communications network planning problem", *Networks 7, 1-23.*

Meyer, R. R. [1985]: "Trust region methods for piecewise-linear approximation", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #626.

Monma, C. L. and Segal, M. [1982]: "A primal algorithm for finding minimum-cost flows in capacitated networks with applications", *Bell System Tech. J. 61, 949-968.*

Moré, J. J. and Sorensen, D. C. [1979]: "On the use of directions of negative curvature in a modified Newton method", *Mathematical Programming, 16, 1-20.*

Murtagh, B. A. and Saunders, M. A. [1978]: "Large-scale linearly constrained optimization", *Mathematical Programming, 14, 41-72.*

Nguyen, S. [1974]: "An algorithm for the traffic assignment problem", *Transportation Science 8, 203-216.*

Nguyen, S. and Dupuis, C. [1984]: "An efficient method for computing traffic equilibria in networks with asymmetric transportation", *Transportation Science 18, 185-202.*

Pang, J. S. and Yu, C. S. [1984]: "Linearized simplicial decomposition methods for computing traffic equilibria on networks", *Networks 14, 427-438.*

Rosenthal, R. E. [1981]: "A nonlinear network flow algorithm for maximization of benefits in a hydroelectric power system", *Operations Res. 29, 763-786.*

Sorensen, D. C. [1982]: "Newton method with a model trust region modification", *SIAM J. Numerical Analysis, 19, 409-426.*

Steenbrink, P. A. [1974]: *Optimization of Transport Network,* Wiley, London.

Vardi, A. [1985]: "A trust region algorithm for equality constrained minimization: convergence properties and implementation" *SIAM J. Numerical Analysis, 22, 575-591.*

Wardrop, J. G. [1952]: "Some Theoretical Aspects of Road Traffic Research" *Proc. Inst. Civil Engr., Part II, 1, 325-378.*

Wolfe, P. [1967]: "Methods of nonlinear programming", in *Nonlinear Programming,* J. Abadie ed.

Zhang, J., Kim, N. H., and Lasdon, L. [1984]: "An improved successive linear programming algorithm", University of Texas Graduate School of Business Working paper 84/85-3-2, Austin.