PARALLEL OPTIMIZATION FOR TRAFFIC ASSIGNMENT

by

R. J. Chen
and
R. R. Meyer

Computer Sciences Technical Report #732

December 1987

PARALLEL OPTIMIZATION FOR TRAFFIC ASSIGNMENT *

R. J. Chen and R. R. Meyer

Computer Sciences Department and Mathematics Research Center

The University of Wisconsin-Madison

Madison, Wisconsin 53706 USA

ABSTRACT

Most large-scale optimization problems exhibit structures that allow the possibility of attack via algorithms that exhibit a high level of parallelism. The emphasis of this paper is the development of parallel optimization algorithms for a class of convex, block-structured problems. Computational experience is cited for some large-scale problems arising from traffic assignment applications. The algorithms considered here 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 in the traffic assignment case, and they may be solved in parallel. Results are given for the distributed solution of such problems on the CRYSTAL multicomputer.

1. INTRODUCTION

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 made inevitable by the goal of achieving the speedups made possible by such architectures.

* This research was supported in part by NSF grant CCR-8709952 and AFOSR grant AFOSR-86-0194.

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 paper, we descibe the very promising results that have been obtained through the use of the CRYSTAL multicomputer on nonlinear multicommodity problems. The methods in essence replace the original optimization problem at each major iteration by an approximation consisting of a set of independent subproblems that may be solved in parallel. After this parallel phase, there is a coordination phase in which results from the subproblems are combined, and the coupling between subproblems is taken into account.

A significant challenge 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 achieve overall parallel efficiency.

In the following section, we describe a class of problems--traffic assignment problems. We then discuss a sequential method in section 3 and a parallel method in section 4. Section 5 concludes this paper with directions for further research. Although the algorithms developed are described in terms of network applications, the theory below is equally applicable to general differentiable convex optimization problems with block-structured constraints.

2. TRAFFIC ASSIGNMENT PROBLEMS

The problem of assignment of traffic to a transportation network arises when 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 routing 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].

2.1. FORMULATION

Consider a transportation network consisting of m nodes and n arcs. A node represents either a zone or an intersection of roads. Two-way traffic 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 $c_j(t_j)$, an optimal traffic pattern satisfying one of Wardrop's principles is a solution to the following problem:

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

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

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

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

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

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 objective function corresponding to a user equilibrium solution [Wardrop 52] is

$$f_j(t_j) = \int_0^{t_j} c_j(x)\, dx \ .$$

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

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

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

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

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 can be formulated as a nonlinear multicommodity problem(MCP):

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

From the block structure of the constraints, it is clear that the approximation of the objective function by a separable function produces an optimization problem that may be decomposed into independent subproblems. The order in which these subproblems are considered leads to a number of different algorithms, which we shall develop after introducing some further terminology.

2.2. NOTATION

For notational convenience, we define

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

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

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

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

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

For a feasible point $\mathbf{x} \in \Omega$, we define a shifted function

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

This function corresponds to the change in the objective function resulting from a change of $\mathbf{d}$ in the current flow $\mathbf{x}$. Thus, for any feasible point $\mathbf{x}$, MCP is equivalent to

$$\min h(\mathbf{d}) \ \ s.t. \ \mathbf{d} \in \Omega_{\mathbf{x}} := \{ \mathbf{d} \mid \mathbf{x}+\mathbf{d} \in \Omega \}. \qquad (MCP_1)$$

We list the notation associated with the shifted function h in the same manner:

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

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

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

We define a *scaled separable function* h^S with scale factor $\sigma > 0$ as

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

When the objective function of (MCP_1) is replaced by h^S, the resulting approximating problem may be decomposed into k independent nonlinear single-commodity subproblems. (The properties of scaled separable functions are discussed in [Chen and Meyer 86]. The introduction of scale factors allows the approximation to be varied in an adaptive manner between an underestimating and over-estimating approximation.) A final approximation step is then done by replacing the separable functions h^S by piecewise-linear approximations h^{PL}, and the resulting piecewise-linear subproblems are solved as linear

single-commodity network problems (for details, see [Chen and Meyer 86]). A number of different algorithms may be developed by varying the scheduling of the solution of these subproblems. At one extreme, the subproblems may be solved sequentially using the most recent updates for the values of the remaining variables. This approach will be termed the block Gauss-Seidel (BGS) method. At the other extreme, all subproblems may be solved in parallel, leading to a method that we will call the parallel block Jacobi (PBJ) method. In between these two extremes there are a variety of hybrids, one of which, the parallel block Gauss-Seidel (PBGS) we will consider in detail below.

3. A SEQUENTIAL METHOD

In this section, we will discuss a sequential algorithm, the block Gauss-Seidel Algorithm(BGS), in which each block of variables is optimized using an approximation based on the most recently updated values for the remaining blocks of variables.

The idea behind BGS is that in each minor iteration we use the most recent values for the other $k-1$ commodities. The philosophy of BGS is thus analogous to that of the Gauss-Seidel algorithm for linear systems. In this context, however, we are dealing with a nonlinear optimization problem, and the algorithm employs a complex approximation process based on the use of trust regions and piecewise linear approximations. The details of these approximation techniques are described fully in [Chen and Meyer 86] and will not be presented here. To help describe 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 : parameter determining size of trust region;

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

LP(α, $\mathbf{x}$) : linearized problem on trust region;

$\mathbf{h}^L$: objective function of linearized problem;

$\mathbf{d}^L$: optimal solution of linearized problem;

$PL_q(\alpha_q, \mathbf{x})$: piecewise-linear problem for commodity q on trust region;

$\mathbf{d}_q$: optimal solution of piecewise-linear problem;

Replacement process with index q :

if $\mathbf{d}_q$ satisfies ratio check (discussed below)

$\mathbf{x}_q \leftarrow \mathbf{x}_q + \mathbf{d}_q$;

else

reduce size of trust region

endif

The block Gauss-Seidel algorithm may now be presented.

BGS Algorithm

step 1: (find a feasible solution)

Find a feasible solution $\mathbf{x}^0$;

initialize indices: $j = 0$ (minor iteration index), $q = 0$;

initialize trust region

step 2: (solve linearized problem periodically)

if $j \in J$

solve LP($\alpha, \mathbf{x}^j$) (linearized problem);

$\theta := h^L(\mathbf{d}^L)$ (optimal value of linearized problem);

endif

increment indices: $q \leftarrow q^+, j \leftarrow j+1$;

step 3: (solve subproblems and do ratio check)

Solve $PL_q(\alpha_q, \mathbf{x}^{j-1})$;

determine if ratio check for commodity q satisfied ($\frac{h_q(\mathbf{d}_q)}{\theta} > \rho_0$);

do *replacement process* for index q;

$\mathbf{x}^j \leftarrow$ result of replacement process;

goto step2

The convergence of BGS is established in [Chen 87].

4. PARALLEL METHODS

In this section, we will discuss parallel algorithms for (MCP), and present numerical results on the Crystal multicomputer. The architecture of the Crystal multicomputer is described in [Feijoo 85].

4.1. A PARALLEL BLOCK JACOBI ALGORITHM

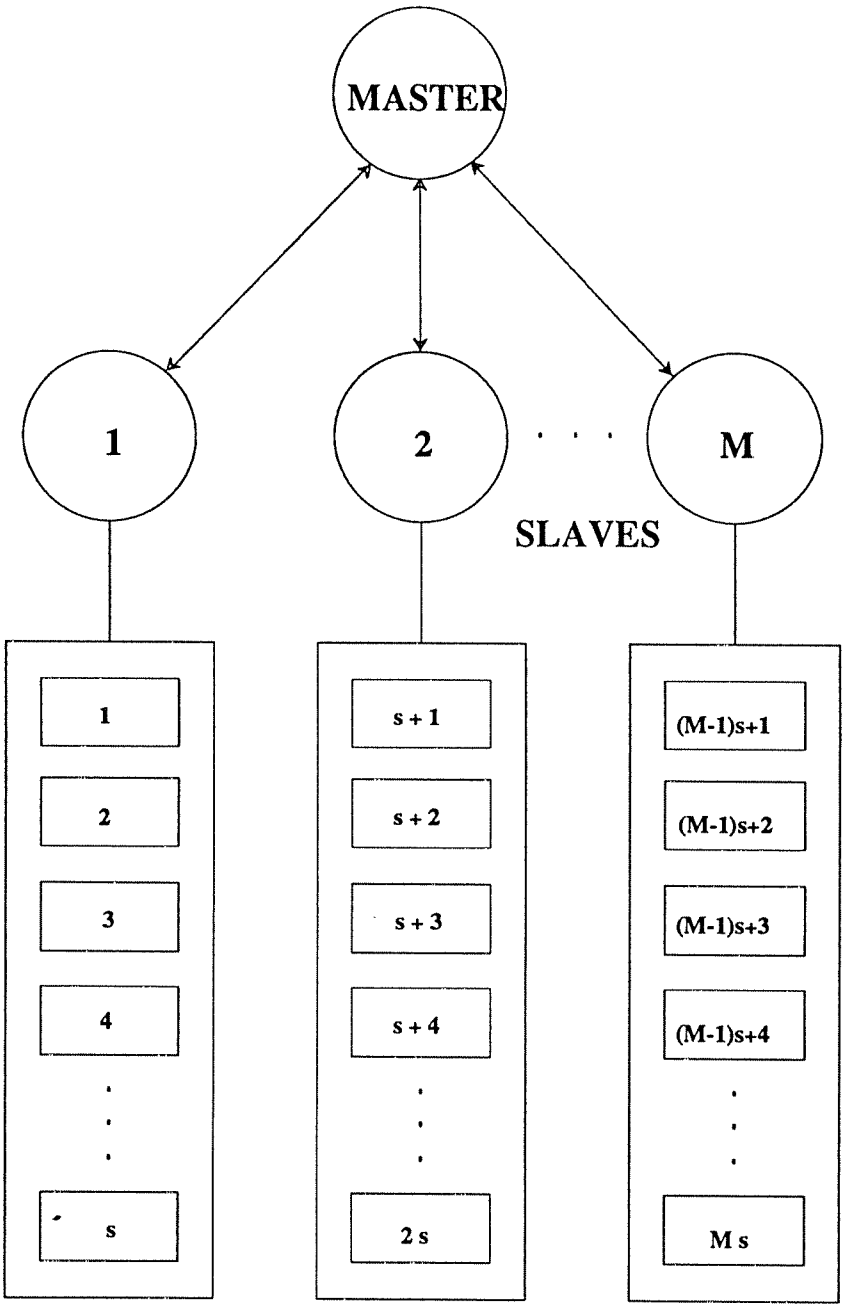
If step 3 of the BGS algorithm is replaced by a step in which all k subproblems are solved in parallel, and the ratio check is done for the sum of the solutions of the piecewise-linear problems, with the updating performed simultaneously for all commodities, then the resulting algorithm is roughly analogous to a block Jacobi algorithm. This procedure and its convergence are discussed in [Chen and Meyer 86]. While this approach is highly parallel, it has the disadvantage that it cannot make use of commodity

update information in the manner of the BGS algorithm, and therefore displays a slower rate of convergence than BGS. In order to achieve parallelism with a better rate of convergence, we therefore develop below an algorithm that may be considered to be a hybrid of BGS and a parallel block Jacobi algorithm.

4.2. A PARALLEL BLOCK GAUSS-SEIDEL ALGORITHM

To increase the efficiency of this decomposition approach, we will develop 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 group of blocks (equal in number to the processors) is updated in parallel (via an optimization procedure applied to each block independently) at each minor 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 (referred to as "master" processor below) 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 (ratio) 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 distribute the s subproblems $1+(p-1)s, 2+(p-1)s, \cdots, ps$ to slave node p, for $p = 1, \cdots, M$ (see Figure 4.1). An example of an execution sequence for PBGS is given in Figure 4.2 below.



k subproblems (k = M s)

Figure 4.1 Distribution of Subproblems on Slave Nodes

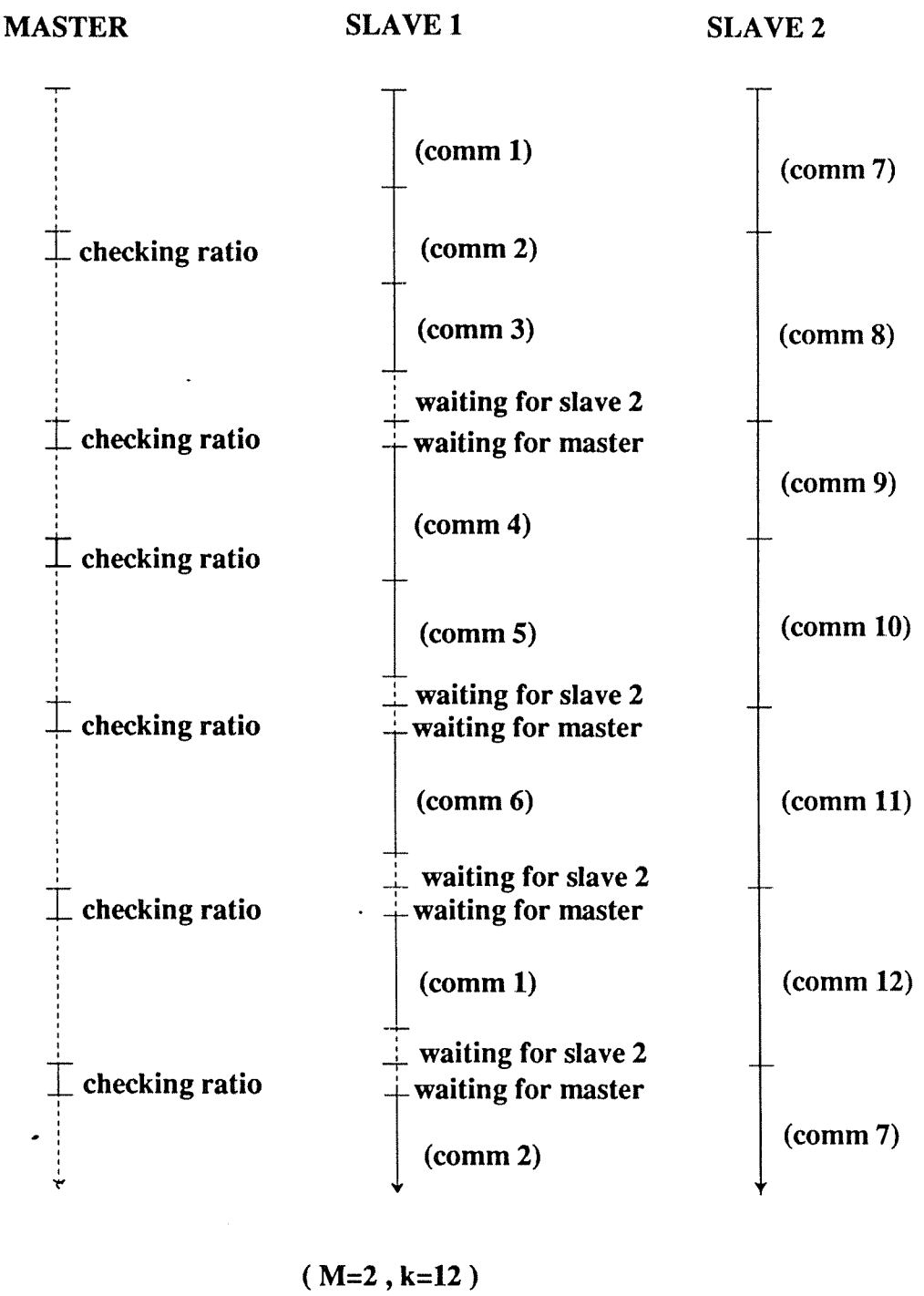


Figure 4.2 An example of an execution sequence for PBGS

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

Notation :

M = # slave machines (each machine solves $s = \frac{k}{M}$ subproblems) ;

$\mathbf{y}_t := (\mathbf{x}_t , \mathbf{x}_{t+s} , .. , \mathbf{x}_{t+(M-1)s})$ ($\mathbf{y}_t$ is *group* of blocks with 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);

$PL_t(\alpha_t , \mathbf{y}) :=$ piecewise-linear problems on trust region for group t ;

j : minor iteration index;

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

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

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

ρ : improvement ratio

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

Replacement process with index t :

if $\mathbf{d}_t$ satisfies ratio check

$\mathbf{y}_t \leftarrow \mathbf{y}_t + \mathbf{d}_t$; (other groups unchanged)

$\alpha_t \leftarrow \max\{\alpha_t , \underline{\alpha}_t\}$; (enforces minimum size of initial trust region)

else

$\alpha_t \leftarrow \gamma \cdot \alpha_t$;

endif

The ratio check cited above means the computation of the ratio of the improvement in the original objective function over the improvement in the linearized objective function and the check that this ratio (to be termed the improvement ratio) is above a fixed threshhold. The ratio check insures that a sufficient decrease in the original objective function is achieved. 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(α, $\mathbf{y}^j$);

$\theta := h^L(\mathbf{d}^L)$;

endif

$t \leftarrow t^+, j \leftarrow j+1$;

step 3: (in parallel solve group t subproblems and do ratio check)

solve (on slaves) $PL_t(\alpha_t, \mathbf{y}^{j-1})$;

perform (on master) ratio check for t^- ($\frac{h_{t^-}(\mathbf{d}_{t^-})}{\theta} > \rho_0$?);

do *replacement process* for index t^-;

$\mathbf{y}^j \leftarrow$ result of replacement process;

goto step2

Remarks

PBGS thus differs from BGS in two important respects: (1) a group of blocks corresponding to M commodities is optimized in parallel at each minor iteration; (2) the replacement process at the current minor iteration is to update the group of blocks for the previous iteration.

4.3. Convergence of PBGS

At each iteration (minor iteration), we check the ratio of the improvement in the original function over the improvement in the linear approximation . The continuity properties of this ratio follow from the assumptions that f is continuously differentiable and convex, and these properties ensure convergence. In the following lemma, $\rho_0 \in (0,\frac{M}{k})$.

Lemma 4.1 If $\mathbf{y}^i \to \bar{\mathbf{y}}$, where $\bar{\mathbf{y}}$ is not a solution of MCP, then there exists a t^* and an $\bar{\alpha} > 0$ such that

$$\frac{h_{t^*}(\mathbf{d}_{t^*}(\alpha))}{h^L(\mathbf{d}^L(\alpha))} \geq \rho_0 \text{ for all } \alpha \in (0,\bar{\alpha}), \text{ and all } \mathbf{y}^i \text{ sufficiently close to } \bar{\mathbf{y}}.$$

Proof: The proof is based on the continuity of the optimal value functions. See [Chen 87] for details. □

Now we will prove the convergence theorem for PBGS.

Theorem 4.2 Any accumulation point generated by PBGS is an optimal solution of MCP.

Proof:

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

Case 1: ($t^* = 1$ satisfies Lemma 4.1.)

We consider those sufficiently large j_i such that $\rho(\mathbf{d}^{j_i},\mathbf{y}^{j_i}) \geq \rho_0$ for all $\alpha \in (0,\bar{\alpha})$. Moreover, since the initial value of α for each distinct $\mathbf{y}^{j_i}$ is at least $\underline{\alpha}$, it is the case that for arbitrarily large j_i that $\alpha^{j_i} \geq \alpha^* := \min\{\gamma\bar{\alpha},\underline{\alpha}\}$ (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 $\theta_0 := h^L(\mathbf{d}^L,\alpha^*,\bar{\mathbf{y}})$, the optimal value of the linearized problem at $\bar{\mathbf{y}}$, we then have

$$h^{j_i}(\mathbf{d}^{j_i}) \leq \rho_0 \cdot \theta^{j_i} \quad (\text{step 3 of PBGS})$$

$$\leq \rho_0 \cdot \frac{\theta_0}{2}, \quad (\text{Lemma 4.4 of [Chen87]})$$

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

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

contradict $f(\mathbf{y}^{j_i}) \to f(\bar{\mathbf{y}})$. So $\bar{\mathbf{y}}$ is a solution. Also $\{f(\mathbf{y}^j)\}$ is a decreasing sequence bounded from below. This implies that any accumulation point of $\mathbf{y}^j$ 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 4.1 there exists some group index t^* for which the improvement ratio is attained. We can choose a subseqence $\{ \mathbf{y}^{j_{i'}} \}$ corresponding to the least such t^* 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. □

4.4. PERFORMANCE FOR PBGS

Test Problems

Five test problems for PBGS are described in the following table.

Name	Comms	Cnstrs	Vars	Obj Fcn
Sioux Falls	24	576	1,824	$\sum a_j t_j^5 + b_j t_j$
Hull	16	6,768	12,768	$\sum t_j a_j (1 + \frac{\alpha_j}{\beta_j + 1} (\frac{t_j}{b_j})^{\beta_j})$
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	$\sum t_j a_j (1 + \frac{\alpha_j}{\beta_j + 1} (\frac{t_j}{b_j})^{\beta_j})$

The Gen problem is a randomly generated problem with a rectangular grid topology . It is typical of the problems produced by the test problem generator described in [Chen 87]. The Sioux Falls, Hull, and Winnipeg problems model the traffic in the respective cities. The Mini-Winnipeg problem is a smaller version that contains only the first 24 commodities of the Winnipeg model.

Convergence rates for PBGS

Intuitively, if we use fewer processors for PBGS we expect to achieve more rapid convergence because the approximating function uses more recent information at each iteration. The following four figures show the rate of convergence using different numbers of processors.

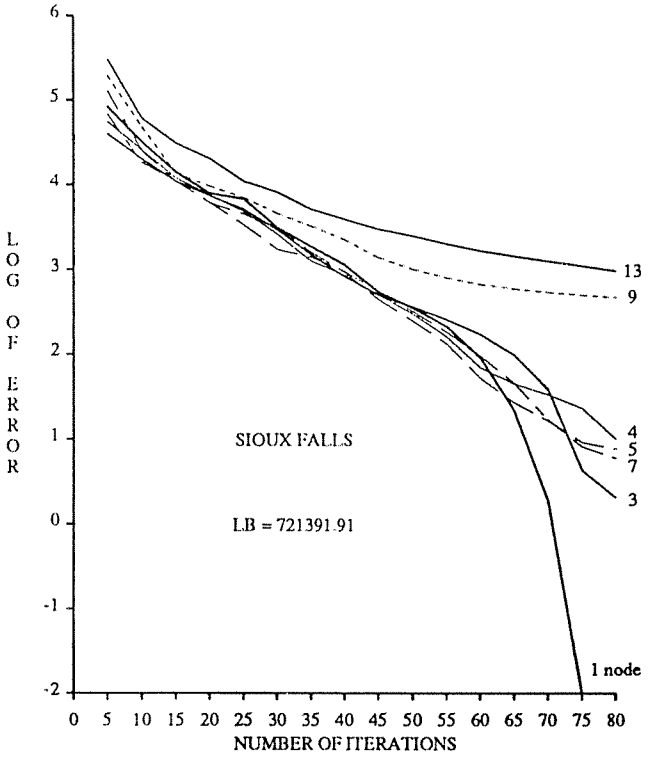


Figure 4.3 Convergence of PBGS (Sioux Falls)

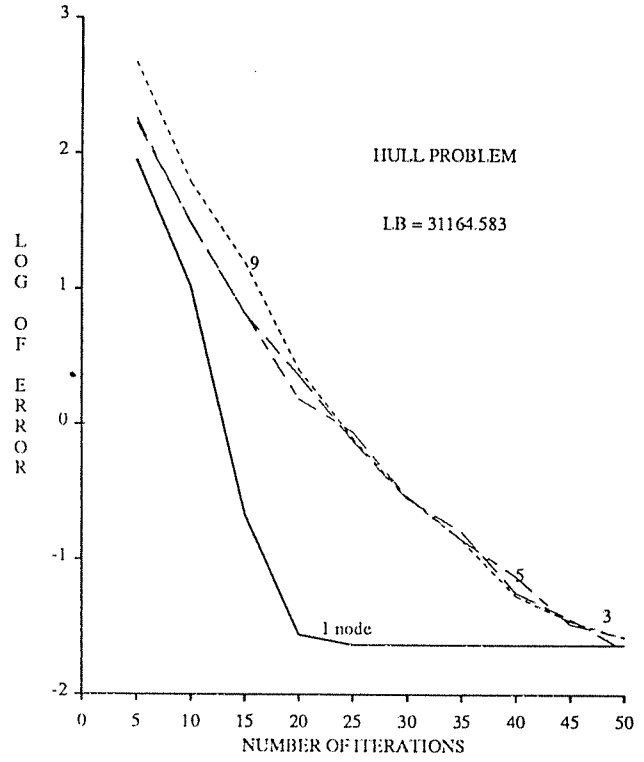


Figure 4.4 Convergence of PBGS (Hull)

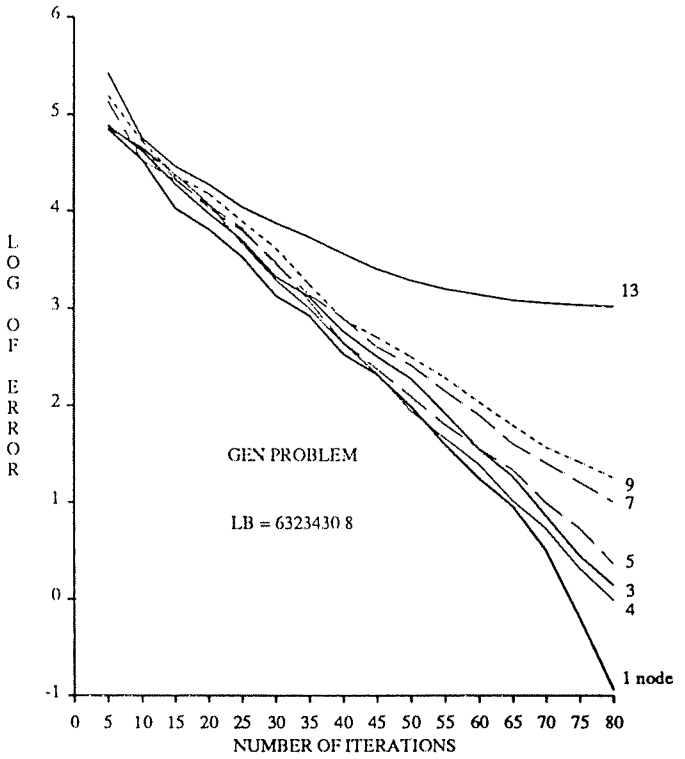


Figure 4.5 Convergence of PBGS (Gen)

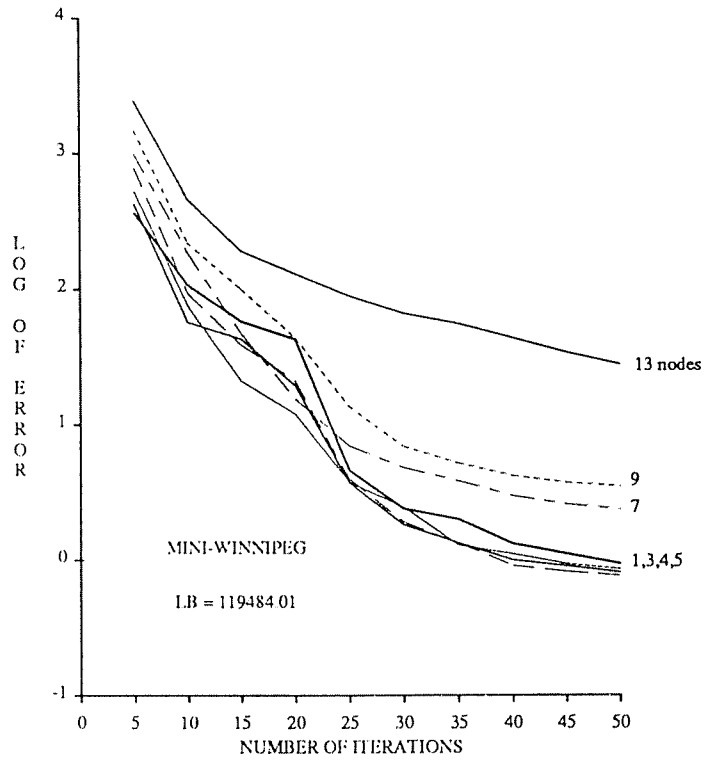


Figure 4.6 Convergence of PBGS (Mini-Winnipeg)

Speedup for PBGS

Setting aside the issue of stopping criteria , we fix the number of iterations and define the speedup of $M+1$ nodes as $\frac{cpu\ time\ \text{for}\ 1\ node}{cpu\ time\ \text{for}\ M+1\ nodes}$. The speedups for the first four test problems are as follows.

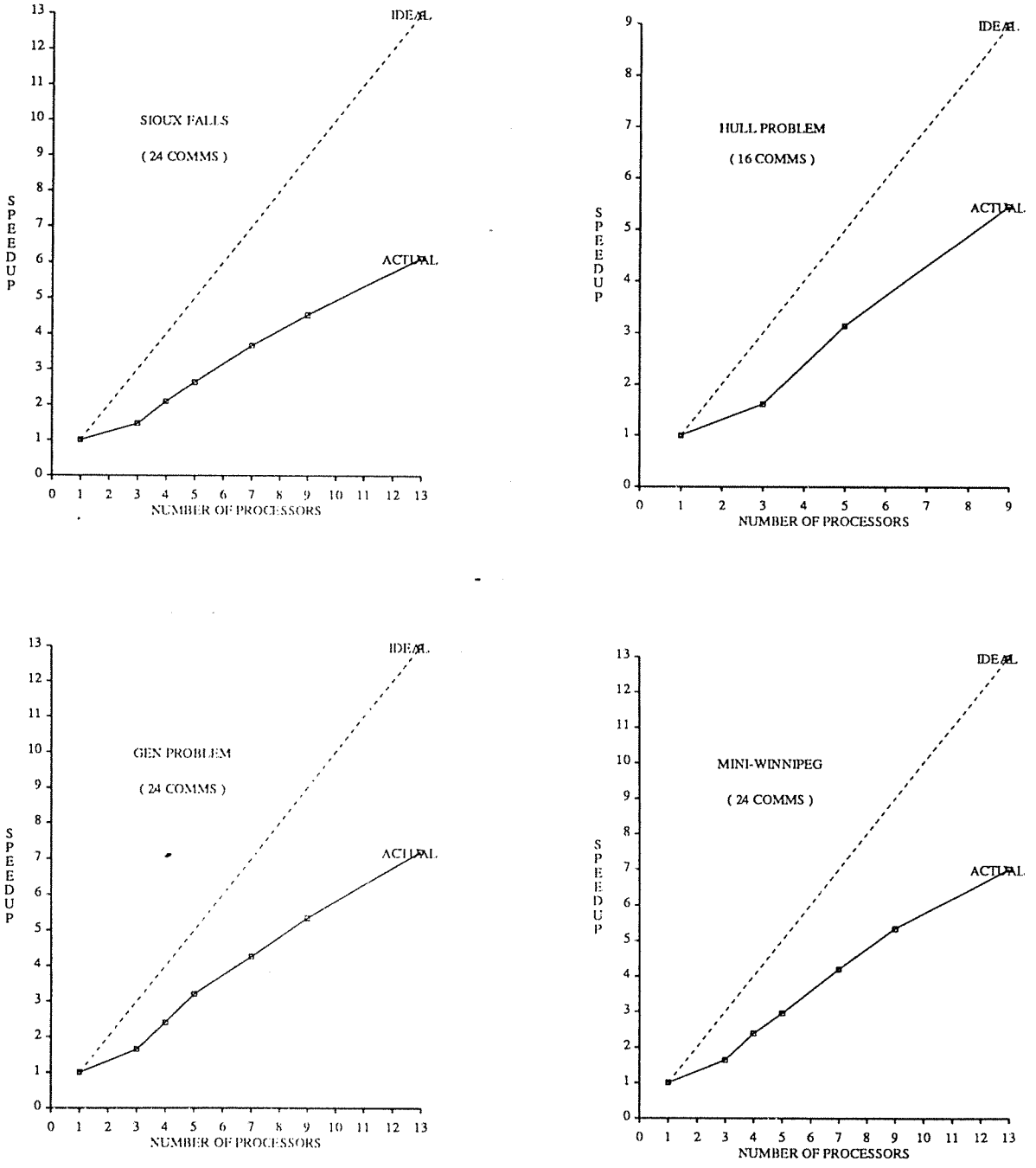


Figure 4.7 Speedup for PBGS

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. 4.8 shows the speedups. A rough lower bound of 622643.35 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

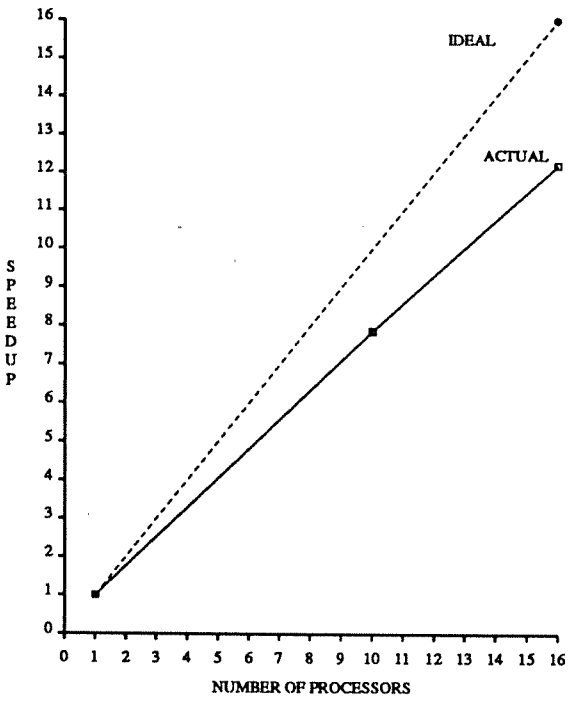


Figure 4.8 Speedup for PBGS (Winnipeg Problem)

5. DIRECTIONS FOR FURTHER RESEARCH

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

5.1. 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 ARPANET is a packet-switched communication network connecting many computer facilities in the United States. A 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 paper to allow both implicit(via the

objective function) and explicit(via nested decomposition) treatment of additional coupling constraints.

5.2. 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. *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.

4. *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 can be implemented as a multitasking program on the Sequent Balance. A parallel scheme using M processors and a global run queue 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 following jobs in parallel (see Fig. 5.1, where an arrow from a processor represents completion of the 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.

Periodically, the processors do the task of solving a linearized problem to obtain an update of the data used in the ratio check.

We are currently investigating this approach on the Sequent as well as algorithms for this problem class that employ multidimensional optimization in the coordination step.

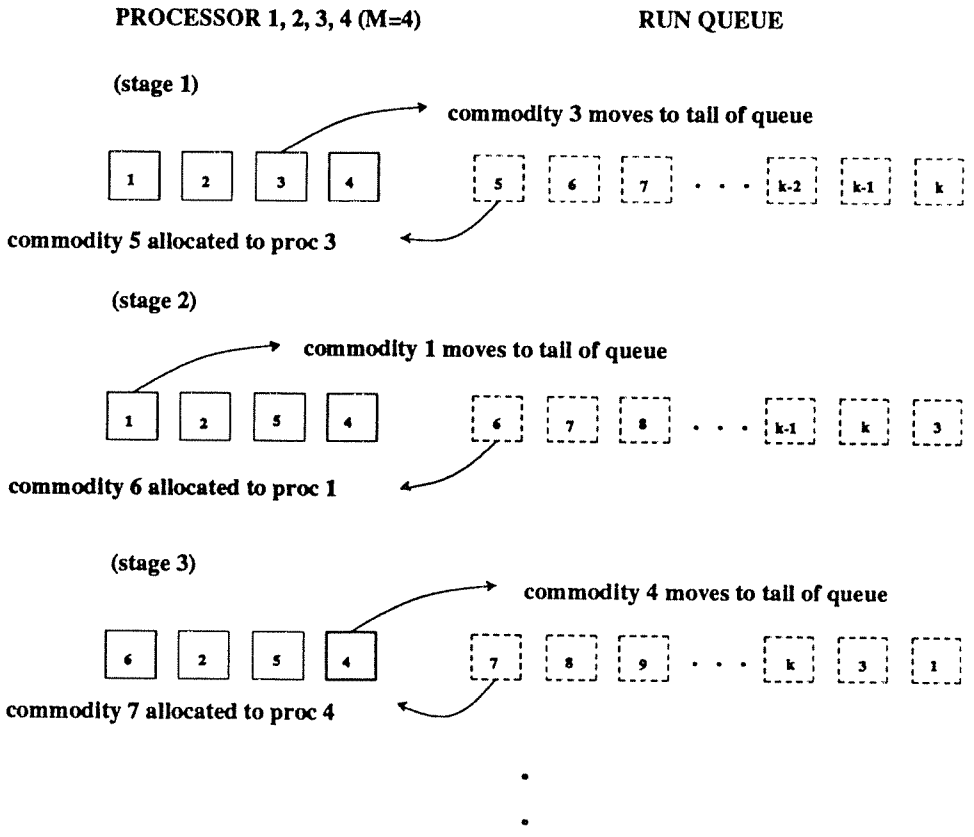


Figure 5.1 A parallel scheme using a global run queue

REFERENCES

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.*

Chen, R.J. : "Parallel algorithms for a class of convex optimization problems", Ph.D. Thesis, Computer Sciences Department, University of Wisconsin-Madison, 1987.

Chen, R. J. and Meyer, R. R. [1986]: "A scaled trust region method for a class of convex optimization problems", University of Wisconsin-Madison Computer Sciences Department Tech. Rpt. #675.

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

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

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.*

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

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

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

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

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.