ASYNCHRONOUS ITERATIVE ALGORITHMS WITH FLEXIBLE COMMUNICATION FOR NONLINEAR NETWORK FLOW PROBLEMS

Didier EL BAZ *, Pierre SPITERI **, Jean Claude MIELLOU *, Didier GAZEN ***

* LAAS du CNRS, L.P. CNRS 8001, 7, avenue du Colonel Roche, 31077 Toulouse Cedex, France, LAASFAX: 61.55.35.77, E-Mail: elbaz@laas.fr

** IRIT - ENSEEIHT, U.A. CNRS 1399, LIMA, Institut National Polytechnique de Toulouse, 2, rue Camichel, 31071 Toulouse Cedex, France.

*** LCS, U.A. CNRS 40741, Université de Franche Comté, 16, Route de Gray, 25030 Besançon Cedex, France.

Abstract: The strictly convex network flow problem is considered. The dual of this problem is unconstrained, differentiable, and well suited for solution via distributed or parallel iterative methods. A new class of asynchronous iterative methods is proposed: the asynchronous iterations with flexible communication. Communication to other processors of the value of the components of the iteration vector resulting from intermediary steps of computation is the main feature of this new class of methods. Convergence is speeded up when such partial updates are used. A convergence result is given. Preliminary computational results are presented and analysed.

Keywords: Distributed algorithms, parallel iterative methods, asynchronous iterations, flexible communication, nonlinear optimization, convex network flow problems.

1. INTRODUCTION

We consider the strictly convex network flow problem. This problem occurs in many domains: electrical networks, gas or water distribution, financial models, communication and transportation networks. Nonlinear network flow problems require intensive computations (see [ZeM88]). A distributed or parallel solution of these problems is very attractive (see [ZeL88], [TsB86], and [BCE95]). We concentrate on the dual problem which is unconstrained, differentiable and well suited for solution via parallel iterative methods. It was shown in [BeE87] and [Elb96] that the structure of the dual problem allows the successful application of distributed asynchronous relaxation and gradient algorithms. Reference is also made to [TsB87], [Tse90], [TBT90], and [ChZ91], for the solution of this problem via sequential or parallel iterative methods using different line search techniques.

In this paper we propose a new class of asynchronous iterative methods, the asynchronous iterations with flexible communication. This new class of algorithms was first presented in [MES94] for the solution of nonlinear systems of equations in the case of M-functions. It was applied in [SME95] to the solution of nonlinear partial differential equations. The general framework of this study is slightly different from the one considered in [MES94] and [SME95]. We consider the case where the nonlinear flow equations are diagonally monotone nondecreasing. This case is more general than M-functions. We also concentrate on a different class of mappings. We study submappings and supermappings, whereas we have considered a-submappings and a-supermappings in [MES94] and [SME95]. Finally, we propose point iterative methods for nonlinear network flow problems, whereas the methods studied in [MES94] and [SME95] are essentially block iterative methods.

Flexible communication between processors is the main feature of the new class of methods presented in this paper. Asynchronous iterations with flexible communication admit more message exchanges than totally asynchronous iterations studied in [ChM69], [Mie75a], [Bau78], and [BeT89]. In particular, the current value of the components of the iteration vector resulting from intermediate steps of updating can be communicated to other processors. We recall that communications occur only at the end of each updating phase in totally asynchronous schemes of computation. So, the new class of methods proposed here allows each processor to communicate partial updates which are issued from computations in progress. We will see in the sequel that the use of such partial updates can speed up the convergence. We give a convergence result which is all new. We note that previous results in the literature (see [ChM69], [Mie75a], [Bau78], and [BeT89]) cannot be used in this context since they consider a different model of algorithms. An implementation of an asynchronous iterative method with flexible communication is proposed. Termination detection is considered. Finally, preliminary computational results on a distributed memory multiprocessor are presented and analysed.

The second Section deals with the convex network flow problem. Asynchronous iterative algorithms with flexible communication are presented in Section three. An implementation is given in Section four. Experimental results are presented and analysed in Section five. Conclusions are drawn in Section six. The proof of the convergence result is presented in Section seven.

2. THE PROBLEM

2.1. Problem formulation

Let $G = (N, A)$ be a connected directed graph. N is referred to as the set of nodes, $A \subset N \times N$ is referred to as the set of arcs, and the cardinal number of N is denoted by n. Let $c_{ij} : R \to (-\infty, +\infty]$ be the cost function associated with each arc $(i,j) \in A, c_{ij}$ is a function of the flow of the arc (i,j) which is denoted by f_{ij}. Let b_i be the supply or demand at node $i \in N$, we have $\sum_{i \in N} b_i = 0$. The problem is to minimize total cost subject to a conservation of flow constraint at each node:

$$min \sum_{(i,j)\in A} c_{ij}(f_{ij}), \text{ subject to } \sum_{(i,j)\in A} f_{ij} - \sum_{(m,i)\in A} f_{mi} = b_i, \forall i \in N. \tag{2.1}$$

We assume that problem (2.1) has a feasible solution. We consider the following standing assumptions on c_{ij}.

Assumption 2.1. c_{ij} is strictly convex.

Assumption 2.2. c_{ij} is lower semicontinuous.

Assumption 2.3. The conjugate convex function of c_{ij}, defined by

$$c^*_{ij}(t_{ij}) = \sup_{f_{ij}}\{t_{ij}.f_{ij} - c_{ij}(f_{ij})\}, \tag{2.2}$$

is real valued, i.e. $-\infty < c^*_{ij}(t_{ij}) < +\infty$ for all real t_{ij}.

Remark 2.1. Assumptions 2.1 to 2.3 correspond to the general assumptions made in [BeE87]. We recall that Assumption 2.3 implies that $\lim_{|f_{ij}|\to\infty} c_{ij}(f_{ij}) = +\infty$. Therefore the objective function of problem (2.1) has bounded level sets (see [Roc70, Section 8]). It follows that there exists an optimal solution for problem (2.1) which must be unique in view of the strict convexity assumed in Assumption 2.1. By the strict convexity of c_{ij}, c^*_{ij} is also continuously differentiable and its gradient denoted by $\nabla c^*_{ij}(t_{ij})$ is the unique f_{ij} attaining the supremum in (2.2) (see [Roc70, pp. 218, 253], see also [BeE87]). We also note that ∇c^*_{ij}, being the gradient of a differentiable convex function, is monotonically nondecreasing.

Problem (2.1) is of great practical interest and has been studied for a long time (see [Roc70] and [Roc84]).

2.2. The dual problem

A dual problem for (2.1) is given by

$$\min_{p\in R^n} q(p), \text{ subject to no constraints on the vector } p = \{p_i | i \in N\}, \tag{2.3}$$

where q is the dual functional given by

$$q(p) = \sum_{(i,j)\in A} c^*_{ij}(p_i - p_j) - \sum_{i\in N} b_i.p_i. \tag{2.4}$$

We refer to p as a price vector and its components as prices. The i-th price p_i, is a Lagrange multiplier associated with the i-th conservation of flow constraint. The duality between problems (2.1) and (2.3) is explored in great detail in [Roc84]. The necessary and sufficient condition for optimality of a pair (f,p) is given in [Roc70]. A feasible flow vector $f = \{f_{ij}|(i,j) \in A\}$ is optimal for (2.1) and a price vector $p = \{p_i|i \in N\}$ is optimal for (2.3) if and only if for all $(i,j) \in A, p_i - p_j$ is a subgradient of c_{ij} at f_{ij}. An equivalent condition is $f_{ij} = \nabla c^*_{ij}(p_i - p_j)$, for all $(i,j) \in A$. Any one of these equivalent relations is referred to as the complementary slackness condition (see [Roc70, pp. 337-338], see also [BeE87]).

2.3. The dual optimal solution set

Existence of an optimal solution of the dual problem can be guaranteed under an additional regular feasibility assumption (see [Roc84, p. 360 and p. 329], see also [BeE87]). On the other hand the optimal solution of the dual problem is never unique since adding the same constant to all coordinates of a price vector p leaves the dual cost unaffected. We can remove this degree of freedom by constraining the price of one node, say a destination node d, to be zero. Consider the set $P = \{p \in R^n | p_d = 0\}$. We concentrate on the reduced dual problem:

$$min_{p \in P} q(p). \tag{2.5}$$

The reduced dual optimal solution set P^* is defined by: $P^* = \{p' \in P | q(p') = \min_p q(p)\}$. In the sequel we will operate under the following assumption.

Assumption 2.4. The reduced dual optimal solution set P^* is nonempty and compact.

Remark 2.2. Assumption 2.4 is not overly restrictive, the reader is referred to [BeE87] for various examples. It follows from Assumption 2.4 that there exists a minimal and a maximal optimal solution of the reduced dual problem, i.e. there exists $\underline{p}, \overline{p} \in P^*$ such that $\underline{p} \leq p \leq \overline{p}$, for all $p \in P^*$, where $\underline{p} \leq p$ denotes the component-wise partial ordering on R^n (see [BeE87, Proposition 2]).

In the sequel $g(p)$ will denote the gradient of the dual functional. From (2.4) it follows that the components $g_i(p)$ of $g(p)$ are given by:

$$g_i(p) = \frac{\partial q(p)}{\partial p_i} = \sum_{(i,j) \in A} \nabla c^*_{ij}(p_i - p_j) - \sum_{(m,i) \in A} \nabla c^*_{mi}(p_m - p_i) - b_i, i \in N. \tag{2.6}$$

Since $g_i(p)$ is a partial derivative of a differentiable convex function it is continuous and monotonically nondecreasing in the i-th coordinate (see [TsB87]).

For simplicity $(\hat{p}_i; p)$ will denote in the sequel the vector of R^n with i-th component equal to $\hat{p}_i$ and j-th component equal to $p_j, j \in N - \{i\}$.

3. PARALLEL ITERATIVE ALGORITHMS

In this Section, we consider several parallel iterative methods for the solution of the reduced dual problem;

in particular, we introduce the new class of asynchronous iterative algorithms with flexible communication.

3.1. Relaxation

Since the reduced dual problem is unconstrained and differentiable it is natural to consider algorithmic solution by a descent iterative method. A relaxation method is interesting in this respect since it admits a simple implementation. Given a price vector $p \in P$, a node $i \in N - \{d\}$ is selected and its price p_i is changed to a value $\hat{p}_i$ such that the dual cost is minimized at $\hat{p}_i$ with respect to the i-th price (i.e. $g_i(\hat{p}_i; p) = 0$), all others prices being kept constant. The algorithm proceeds by relaxing the prices of all nodes element of $N - \{d\}$ in cyclic order and repeating the process (see [BeE87]). Consider now the point-to-set mapping $F_i, i \in N - \{d\}$, which assigns to a price vector $p \in P$, the set of all prices $\hat{p}_i$ that minimize the dual cost along the i-th price starting from p, i.e. $F_i(p) = \{\hat{p}_i | g_i(\hat{p}_i; p) = 0\}$. It is well known that a real valued convex function having one compact level set, has all its level sets compact (see [Roc70 p.70]). Therefore under Assumption 2.4 the sets $F_i(p), p \in P, i \in N - \{d\}$, are all nonempty, compact intervals. It follows that the minimal and maximal relaxation mappings $\underline{F}$ and $\overline{F}$, with components defined by:

$$\underline{F}_i(p) = \min_{\hat{p}_i \in F_i(p)} \hat{p}_i \text{ and } \overline{F}_i(p) = \max_{\hat{p}_i \in F_i(p)} \hat{p}_i, i \in N - \{d\}, \tag{3.1}$$

respectively, are well defined on the set P (see [BeE87]). Bertsekas and El Baz have shown in [BeE87] that $\underline{F}$ and $\overline{F}$ are continuous and monotone on P in the sense that for any $p, p' \in P$, $i \in N - \{d\}$ we have

$$\underline{F}_i(p) \leq \underline{F}_i(p') \text{ if } p \leq p',$$

$$\overline{F}_i(p) \leq \overline{F}_i(p') \text{ if } p \leq p'.$$

The following two concepts will be central to all our considerations.

3.2. Submappings and supermappings

Definition 3.1. A mapping $\check{F}$, with components $\check{F}_i, i \in N - \{d\}$, is a submapping associated with the minimal relaxation mapping $\underline{F}$ on $P' = \{p \in P | p \leq \underline{p}\}$, we also say that $\check{F}$ is a submapping on P' for simplicity, if for all $i \in N - \{d\}$ and $p \in P'$ such that $p_i \leq \underline{F}_i(p)$, we have $\check{F}_i(p) \in [p_i, \underline{F}_i(p)]$ and $\check{F}_i(p) \neq p_i$ if $\underline{F}_i(p) \neq p_i$.

We can analogously define supermappings on $P" = \{p \in P | \overline{p} \leq p\}$, by substituting $\overline{F}$ for $\underline{F}$, $P"$ for P', and reversing the inequalities.

Remark 3.1. By the monotonicity of $\underline{F}$ on P, (3.1) implies that if $p \in P'$, then we have $\underline{F}_i(p) \leq \underline{F}_i(\underline{p}) = \underline{p}_i$, for all $i \in N - \{d\}$. Therefore, it follows from Definition 3.1 that if $\check{F}$ is a submapping on P', then for all $i \in N - \{d\}$ and $p \in P'$ such that $p_i \leq \underline{F}_i(p)$, we have $\check{F}_i(p) \leq \underline{F}_i(p) \leq \underline{p}_i$. We can analogously show that if $\check{F}$ is a supermapping on $P"$, then for all $i \in N - \{d\}$ and $p \in P"$ such that $\overline{F}_i(p) \leq p_i$, we have $\overline{p}_i \leq \check{F}_i(p)$.

Now we introduce properties referring to continuity. These properties will be used in the convergence analysis of asynchronous iterative algorithms with flexible communication.

Definition 3.2. A submapping $\check{F}$ is order continuous on P' if

$$p(k) \uparrow p' \in P', k \to \infty \Rightarrow \check{F}_i(p(k)) \uparrow \check{F}_i(p'), k \to \infty, \text{ for all } i \in N - \{d\}, \tag{3.2}$$

where the notation $p(k) \uparrow p', k \to \infty$, is used to mean that $p(0) \leq p(1) \leq ... \leq p(k) \leq p(k+1) \leq ... \leq p'$ and $\lim_{k\to\infty} p(k) = p'$.

A supermapping $\check{F}$ is order continuous on P" if

$$p(k) \downarrow p' \in P\text{''}, k \to \infty \Rightarrow \check{F}_i(p(k)) \downarrow \check{F}_i(p'), k \to \infty, \text{ for all } i \in N - \{d\},$$

where the notation $p(k) \downarrow p', k \to \infty$, is used to mean that $p(0) \geq p(1) \geq ... \geq p(k) \geq p(k+1) \geq ... \geq p'$ and $\lim_{k\to\infty} p(k) = p'$.

Definition 3.3. Let $\check{F}$ be a submapping on P' and $\check{F}'$ an order continuous submapping on P' such that for all $i \in N - \{d\}$ and $p \in P'$, satisfying $p_i \leq \underline{F}_i(p)$, we have

$$\check{F}'_i(p) \in [p_i, \check{F}_i(p)], \tag{3.3}$$

then the submapping $\check{F}$ is m-continuous on P'.

We can similarly define m-continuous supermappings on P" by substituting P" for P' and reversing the inequalities.

3.3. Examples of submappings and supermappings

Now we present several submappings and supermappings. We consider first mappings based on inexact line search.

3.3.1. Relaxation methods with inexact line search

In this subsection, we concentrate on approximate relaxation mappings $\tilde{F}$ whereby the minimization along each coordinate is allowed to be inexact to some extent. Reference is made in particular to [TsB87], [BHT87], [BeT89], [Tse90], and [ChZ91]. For all $p \in P$, the components $\tilde{F}_i, i \in N - \{d\}$, of the mapping $\tilde{F}$, are defined by:

$$\delta g_i(p) \leq g_i(\tilde{F}_i(p);p) \leq 0 \text{ and } \tilde{F}_i(p) \leq \underline{F}_i(p), \text{ if } g_i(p) < 0, \tilde{F}_i(p) = p_i, \text{ if } g_i(p) = 0, \tag{3.4a}$$

$$0 \leq g_i(\tilde{F}_i(p);p) \leq \delta g_i(p) \text{ and } \tilde{F}_i(p) \geq \overline{F}_i(p), \text{ if } g_i(p) > 0, \tag{3.4b}$$

where $\delta \in [0,1)$.

Proposition 3.1. Let Assumptions 2.1 to 2.4 hold. Then, any mapping $\tilde{F}$, defined by (3.4) is a submapping on P'.

Proof: For all $i \in N - \{d\}$ and $p \in P'$ such that $p_i \leq \underline{F}_i(p)$, we have $g_i(p) \leq g_i(\underline{F}_i(p);p) = 0$, since g_i is monotonically nondecreasing in the i-th coordinate, moreover it follows from (3.1) that if $\underline{F}_i(p) \neq p_i$, then we have $g_i(p) < 0$. Hence, it follows from (3.4a) that $p_i < \tilde{F}_i(p) \leq \underline{F}_i(p)$. If $p_i = \underline{F}_i(p)$, then we have $g_i(p) = 0$ and it follows from (3.4a) that $\tilde{F}_i(p) = p_i$.

Q.E.D.

It can analogously be shown that any mapping $\tilde{F}$, defined by (3.4) is also a supermapping on $P" = \{p \in P | \overline{p} \leq p\}$. We note that the converse of Proposition 3.1 is not true. Clearly, all submappings on P' do not satisfy (3.4a).

In some cases the following assumption is introduced.

Assumption 3.1. c_{ij} is continuously differentiable.

The differentiability assumption is made in particular in [ChZ91], in that case the mapping $\tilde{F}$ considered is continuous. Therefore, $\tilde{F}$ is clearly order continuous.

3.3.2. Gradient and related mappings

In this subsection, we present several submappings and supermappings which are related to the gradient mapping. The components F'_i of the gradient mapping F' are defined by

$$F'_i(p) = p_i - \frac{1}{\alpha} g_i(p), \text{ for all } i \in N - \{d\} \text{ and } p \in P. \tag{3.5}$$

where α is a positive constant. Clearly F' is continuous since g is continuous. We introduce the following assumption.

Assumption 3.2. c_{ij} is strongly convex with modulus $\frac{1}{\beta}$.

It was shown in [Elb96, Theorem 2.2] that under Assumptions 2.1 to 2.4 and Assumption 3.2, there exists a constant $\alpha = \beta . \max_{i \in N} a_i$, where a_i denotes the degree of node $i \in N$, such that for all $p, p' \in P$ satisfying $p' \leq p$, we have: $g(p) - g(p') \leq \alpha .(p - p')$. Therefore, the gradient mapping F' is monotone on P if $\alpha = \beta . \max_{i \in N} a_i$. It can also be shown that F' is monotone on a given adequate subset of P' if c_{ij} is strongly convex on an associated subdomain (see [Elb96]). The reader is referred to [Elb96, p. 195] for the presentation of monotone scaled gradient mappings F', with components F'_i given by $F'_i(p) = p_i - \frac{1}{\alpha_i} g_i(p)$, for all $i \in N - \{d\}$ and $p \in P$, where $\alpha_i = \beta . a_i$.

Proposition 3.2. Let Assumptions 2.1 to 2.4 and 3.2 hold and $\alpha = \beta . \max_{i \in N} a_i$. Then the gradient mapping F' is an order continuous submapping on P'.

Proof: For all $i \in N - \{d\}$ and $p \in P'$ such that $p_i \leq \underline{F}_i(p)$, we have $g_i(p) \leq g_i(\underline{F}_i(p); p) = 0$, since g_i is monotonically nondecreasing in the i-th coordinate. Therefore it follows from (3.5) that $p_i \leq F'_i(p)$. It follows from the monotonicity of F' on P', (3.5), and (3.1) that for all $i \in N - \{d\}$ and $p \in P'$ such that $p_i \leq \underline{F}_i(p)$, we have

$$F'_i(p) \leq F'_i(\underline{F}_i(p), p) = \underline{F}_i(p) - \frac{1}{\alpha} g_i(\underline{F}_i(p); p) = \underline{F}_i(p).$$

Moreover it follows from (3.1) and (3.5) that $F'_i(p) \neq p_i$, if $\underline{F}_i(p) \neq p_i$. Finally, the gradient mapping F' is order continuous on P' since F' is continuous on P'.

Q.E.D.

We can analogously show that the gradient mapping F' is also an order continuous supermapping on P".

Now we extend the class of submappings and supermappings related to the gradient mapping.

Proposition 3.3. Let the assumptions of Proposition 3.2 hold. Then the mapping $\check{F}$ defined by: $\check{F}_i(p) = p_i^{q'}$, for all $i \in N - \{d\}$ and $p \in P$, where q' is a given constant and $p_i^q = F'_i(p_i^{q-1}; p), q = 1, ..., q', p_i^0 = p_i$, is an m-continuous submapping on P'.

Proof: We show that $\check{F}$ is a submapping minorized by the order continuous gradient submapping F' on P'.

We argue by induction. For $q' = 1$, the mapping $\check{F}$ is the gradient mapping and the proof of Proposition 3.2 referring to the gradient mapping F', gives the first step of the induction. Assume now that there exists $q, 1 \leq q < q'$, such that for all $i \in N - \{d\}$ and $p \in P'$ with $p_i \leq \underline{F}_i(p)$, we have

$$p_i^0 \leq ... \leq p_i^{q-1} \leq p_i^q \leq \underline{F}_i(p) \text{ and } p_i^q \neq p_i \text{ if } \underline{F}_i(p) \neq p_i.$$

Hence, by the monotonicity of F' on P' it follows from (3.1) that

$$p_i^{q+1} = F_i'(p_i^q; p) \geq F_i'(p_i^{q-1}; p) = p_i^q \text{ and}$$

$$p_i^{q+1} = F_i'(p_i^q; p) \leq F_i'(\underline{F}_i(p), p) = \underline{F}_i(p) - \frac{1}{\alpha} g_i(\underline{F}_i(p); p) = \underline{F}_i(p).$$

Finally, $p_i^{q+1} \neq p_i$ if $\underline{F}_i(p) \neq p_i$ since $p_i^{q+1} \geq p_i^q \geq p_i$ and $p_i^q \neq p_i$ if $\underline{F}_i(p) \neq p_i$.

Q.E.D.

Remark 3.2. Clearly, the same result holds for the mapping $\check{F}$ defined by: $\check{F}_i(p) = p_i^{q'}$, for all $i \in N - \{d\}$ and $p \in P$, where $p_i^q = F_i'(p_i^{q-1}; p), q = 1, ..., q', p_i^0 = p_i$, and q' is the smaller positive integer such that $\left| g_i(p_i^{q'}; p) \right| \leq \epsilon'$, ϵ' being a given constant.

We can show analogously that the above mappings $\check{F}$ are m-continuous supermappings on P".

The concepts of submapping and supermapping are general. Concepts referring to submappings and supermappings have been introduced previously in the context of the solution of nonlinear systems of equations involving M-functions (see [Mie86]). Clearly these concepts can also be introduced for the solution of minimization problems with strictly convex separable cost and linear constraints.

3.4. Asynchronous iterative algorithms with flexible communication

It follows from equation (2.6) that only local data, i.e. prices of adjacent nodes, are needed to update a price. Relaxation or iterative methods associated with a submapping or a supermapping can be implemented in a distributed way or in parallel. Prices p_i of a subset of nodes can be updated concurrently by several processors. For example prices of nodes that are not directly connected can be updated simultaneously in the case of relaxation methods. All prices can be updated simultaneously in the case of the gradient method. These implementations are carried out according to a particular order, moreover they require synchronization. Since idle time due to synchronization may be nonnegligible, the parallel implementation of the above iterative methods may be improved by considering procedures whereby computations are carried out concurrently without any order nor synchronization, namely asynchronous procedures. The restrictions imposed on asynchronous iterative methods are very weak: no component of the iteration vector is abandoned forever and more and more recent values of the components have to be used as the computation progresses. For further details about asynchronous iterative algorithms the reader is referred to [ChM69], [Mie75a], [Bau78], and [BeT89].

Convergence of asynchronous iterative algorithms has been established for many problems (see [Bau78] - [BeT89], [ChM69] - [Elb91], [Elb94a] - [MiS85b], and [TsB86] - [UrD90]). Particular attention must be paid to the Asynchronous Convergence Theorem of Bertsekas (see [Ber83], see also [BeT89, p. 431]). For network flow problems, Bertsekas and El Baz have shown the convergence of asynchronous relaxation methods under Assumptions 2.1 to 2.4 (see [BeE87]). Satisfactory convergence properties of asynchronous gradient algorithms were also shown in [Elb96] using in particular Assumptions 2.1 to 2.3 and 3.2.

The asynchronous model of computation does not allow the communication of the results of intermediate steps of updating; it can be interesting to communicate both updates and such partial updates to other processors. So, we propose a new approach to constructing asynchronous iterative algorithms. We introduce the class of asynchronous iterations with flexible communication. This new class of asynchronous iterative algorithms was first proposed in [MES94] in the context of M-functions and a-submappings.

Asynchronous iterations with flexible communication are associated here with fixed point mappings which are submappings or supermappings and present the property of being generated by an iterative process like the gradient type mappings of Proposition 3.3 and Remark 3.2.

Definition 3.4. An asynchronous iteration with flexible communication associated with an m-continuous submapping $\check{F}$ on P' and the starting point $p(0) \in P'$, satisfying $p(0) \leq \underline{F}(p(0))$, is a sequence $\{p(k)\}$ of vectors of P such that for $i \in N - \{d\}$:

$$p_i(k) \in [\check{F}_i'(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))), \check{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k)))], \text{ for all } k \in T^i, \tag{3.6}$$

$$p_i(k) = p_i(k-1), \text{ for all } k \notin T^i,$$

where $\check{F}'$ is an order continuous submapping which minorizes $\check{F}$, $T = \{0, 1, 2, ...\}$ denotes the set of times at which the current value of a component of the iteration vector is communicated to an other processor, T^i is the subset of times at which the current value of the i-th component of vector p is communicated by a processor to an other processor, and for all $i \in N - \{d\}$:

the set T^i is infinite, (3.7)

$0 \leq \tau_j^i(k) \leq k - 1$, for all $j \in N - \{i, d\}$ and $k \in T^i$, (3.8)

$\tau_i^i(k) = k - 1$, for all $k \in T^i$, (3.9)

$\tau_j^i(k)$ is monotonically increasing for all $j \in N - \{d\}$, (3.10)

if $\{k_t\}$ is a sequence of elements of T^i that tends to infinity, then $\lim_{t\to\infty} \tau_j^i(k_t) = +\infty$ for all j. (3.11)

We similarly define asynchronous iterative algorithms with flexible communication associated with an m-continuous supermapping $\check{F}$ on P" and a starting point $p(0) \in P$" such that $p(0) \geq \overline{F}(p(0))$, by substituting the following expression for (3.6):

$$p_i(k) \in [\check{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))), \check{F}_i'(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k)))], \text{ for all } k \in T^i,$$

$$p_i(k) = p_i(k-1), \text{ for all } k \notin T^i.$$

According to (3.7) to (3.11) no component of the iteration vector is abandoned for ever, the latest value of p_i is used at each computation of p_i, finally more and more recent values are used as the computation progresses. Asynchronous iterations with flexible communication defined by (3.6) to (3.11) describe general iterative methods whereby computations are carried out in parallel by up to $n-1$ processors without any order nor synchronization. The main feature of this new class of algorithms is flexible communication between processors. In this new model, the prices $p_i(k)$ which are communicated to other processors can correspond to new updates of p_i produced by the submapping $\check{F}$ or to the current value of p_i produced by only few steps of the iterative process which generates the submapping $\check{F}$, a restriction being that $p_i(k)$ takes value between $\check{F}_i'(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k)))$ and $\check{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k)))$. So, the values $p_i(k)$ delivered by the algorithm can correspond to updates or partial updates issued from computations which are in progress. We also note that the values $p_j(\tau_j^i(k))$ used in the updating of p_i are upperbounded by $p_j(k-1)$, the current value of p_j at time $k-1$.

The above model describes general asynchronous schemes of computation whereby each processor can have access to the current state of other processors. So, asynchronous iterations with flexible communication admit more data exchange between processors than totally asynchronous iterations studied in [ChM69], [Mie75a], [Bau78], [Ber83], and [BeT89] since the current value of each component of the iteration vector can be communicated to other processors at any time and possibly without any fixed rule; we recall that communications occur only at the end of each updating phase in the totally asynchronous iterative scheme of computation. We also note that asynchronous iterations with flexible communication are tightly bound to the concepts of submappings and supermappings which are associated with the generation of monotone sequence of vectors; this is the reason why the supplementary assumption (3.10), which is dropped in the Asynchronous Convergence

Theorem of Bertsekas (see [Ber83], and [BeT89, p. 431]), is needed here and the convergence mechanism of asynchronous iterations with flexible communication, presented in detail in the Appendix, is different from the convergence result presented in [Ber83], and [BeT89, p. 431]. The use of the resulting values of intermediary steps of updating presents a particular interest in this context; intuitively, it can speed up the convergence.

Finally, we note that there are various ways of implementing asynchronous iterations with flexible communication. In the sequel, we shall briefly quote two ways. The first manner corresponds to the case where each processor sends a request to other processors upon beginning a new updating and receives their current state which can correspond to an intermediary step of computation. Another method corresponds to the case where each processor transmits its current state to other processors according to a given policy. The latter method is detailed in subsection 4.3.

The following result states the monotone convergence of asynchronous iterations with flexible communication.

Proposition 3.4. Let $\check{F}$ be an m-continuous submapping on P', $p(0) \in P'$ a starting point such that $p(0) \leq \underline{F}(p(0))$, and $\underline{p}$ the minimal optimal solution of the problem. Then, asynchronous iterations with flexible communication $\{p(k)\}$ associated with $\check{F}$ and $p(0)$ are well defined and satisfy $p(k) \uparrow \underline{p}, k \to \infty$.

Proof: see Appendix.

Remark 3.3. We can analogously show that asynchronous iterative methods with flexible communication associated with an m-continuous supermapping $\check{F}$ on P" converge to the maximal solution of the problem $\overline{p}$, from $p(0) \in P$" such that $p(0) \geq \overline{F}(p(0))$.

4. IMPLEMENTATION

The implementation was carried out in parallel C on a T-node 16-32 distributed memory multiprocessor. The machine consists of a network of 16 to 32 T800 transputers with some local memory. A T800 transputer is a chip that integrates a processor, a floating point unit, fast memory and four bidirectional communication links. The processor, floating point unit, and memory make the chip suitable as a building component for computers. The communication links allow more transputers to be connected into one multiprocessor configuration. Communication is made via direct memory access. Various network topologies can be programmed via an Inmos C004 crossbar: pipeline, ring, grid, cube... In this study, we consider a pipeline network of processors with bidirectional links (see Figure 1).

4.1. Synchronous implementation

In the case of synchronous iterative algorithms, updating and data exchange are made sequentially. Processors communicate the updates at the end of each updating phase. Communication only occurs with adjacent processors in the pipeline. Processors are synchronized by message exchange, since communication is synchronized and not bufferised.

4.2. Asynchronous implementation

Since only synchronous communication facilities were provided, we have considered the following implementation of asynchronous iterations. Two concurrent processes are implemented in parallel on each processor: a computation process performs updatings and sends the updates to adjacent processors, a buffer process receives and stores data sent by adjacent processors. The use of the buffer process allows the implementation of asynchronous communications and more generally of asynchronous algorithms. The buffer process which has very fast elementary subprocesses has a higher level of priority than the computation process which consumes more time. So, the receipt of data is not delayed. The buffer process is idle while waiting for messages. All the cpu time is then allocated to the process computation since the scheduler of the T800 transputer is designed so that idle processes do not consume cpu time.

The approach considered in this study differs from the one considered in [Elb93]; as a matter of fact the computation and the buffer processes communicate via shared variables rather than via message passing (see Figure 1). Therefore, the computation process does not send requests to the buffer process. Moreover the computation process does not wait for a reply. So, the buffer process is fully devoted to the receipt and storage of external data. Thus message exchanges between processors are improved. We note that the data stored in the shared variables only correspond to the value of the components of the iteration vector communicated by other processors. The computation process reads the latest available data in the shared variables at the beginning of each new updating.

4.3. An implementation of asynchronous iterations with flexible communication

We propose an implementation of asynchronous iterations with flexible communication which presents many similarities with the implementation of asynchronous iterations given in subsection 4.2: two concurrent processes run in parallel in each processor: a low level priority computation process and a high level priority buffer process. The computation and buffer processes communicate via shared variables. In addition to the communications made in the asynchronous implementation presented in subsection 4.2, the current value of the components of the iteration vector is also communicated to other processors when a given number of intermediate steps of the updating process is reached. More precisely, the submapping of Remark 3.2 is implemented, and the current value of the components of the iteration vector is sent when q is equal to a given integer.

We illustrate the implementation by an algorithm. In order to simplify the presentation, we consider the simple case where each processor updates one price.

PAR $i = 1$ FOR $n - 1$

    PAR

        BUFFER

        COMPUTATION(i)

In the process COMPUTATION(i), the following process updates p_i :

read shared variables

$q \leftarrow 0$

$p_i^q \leftarrow (p_i)$

WHILE $|g_i(p_i^q; p)| \geq \epsilon$ THEN

$p_i^{q+1} \leftarrow p_i^q - \frac{1}{\alpha} g_i(p_i^q; p)$

$q \leftarrow q + 1$

IF $q \in Q$ THEN communicate the current value of p_i (i.e. p_i^q) to adjacent processors

read shared variables

$p_i^{q+1} \leftarrow p_i^q - \frac{1}{\alpha} g_i(p_i^q; p)$

communicate the update p_i^{q+1} to adjacent processors

where Q is a set of given positive integers.

4.3.1. Termination test

The termination test is based on the value of $\left|g_d(p_1(\tau_1^d(k)), ..., p_n(\tau_n^d(k)))\right|$; T^d denotes the infinite subset of times at which the termination test is performed and we assume that the $\tau_j^d(k))$ satisfy (3.8), (3.10), and (3.11). We show that this test can be used to detect the global termination of asynchronous iterations with flexible communication.

Proposition 4.1. Let Assumptions 2.1 to 2.4 hold and $\{p(k)\}$ be an asynchronous iterative algorithm with flexible communication associated with an m-continuous submapping $\check{F}$ on P' starting from $p(0) \in P'$ such that $p(0) \leq \underline{F}(p(0))$. Then $g_d(p(k)) \geq 0$, for all k.

Proof: It follows from the proof of Proposition 3.4 that $p(k) \in P' = \{p \in P | p \leq \underline{p}\}$, for all k. Therefore for all $(m, d), (d, j) \in A$, we have

$$p_m(k) - p_d(k) \leq \underline{p}_m - \underline{p}_d \text{ and } p_d(k) - p_j(k) \geq \underline{p}_d - \underline{p}_j,$$

where $p_d(k) = \underline{p}_d = 0$.

By the monotonically nondecreasing property of ∇c_{ij}^*, for all $(m, d), (d, j) \in A$, we have

$$\nabla c_{md}^*(p_m(k) - p_d(k)) \leq \nabla c_{md}^*(\underline{p}_m - \underline{p}_d) \text{ and } \nabla c_{dj}^*(p_d(k) - p_j(k)) \geq \nabla c_{dj}^*(\underline{p}_d - \underline{p}_j).$$

It follows from (2.6) that $g_d(p(k)) \geq g_d(\underline{p}) = 0$, for all k.

Q.E.D.

Proposition 4.2. Let the assumptions of Proposition 4.1 hold, ϵ be a positive constant, and assume that there exists k such that $g_d(p(k)) \leq \epsilon$. Then $g_d(p(k')) \leq \epsilon$, for all $k' \geq k$.

Proof: It follows from the proof of Proposition 3.4 that the asynchronous iteration with flexible communication $\{p(k)\}$ satifies:

$$p(k) \leq p(k'), \text{ for all } k' \geq k. \tag{4.1}$$

By the monotonically nondecreasing property of ∇c^*_{ij}, for all $(m,d),(d,j) \in A$, we have

$$\nabla c^*_{md}(p_m(k) - p_d(k)) \leq \nabla c^*_{md}(p_m(k') - p_d(k')) \text{ and } \nabla c^*_{dj}(p_d(k) - p_j(k)) \geq \nabla c^*_{dj}(p_d(k') - p_j(k')).$$

Hence, it follows from (2.6) that $g_d(p(k')) \leq g_d(p(k)) \leq \epsilon$, for all $k' \geq k$.

Q.E.D.

The following proposition is the main result of this subsection.

Proposition 4.3. Let the assumptions of Proposition 4.1 hold, ϵ be a positive constant, and assume that there exists k such that $g_d(p_1(\tau_1^d(k)), ..., p_n(\tau_n^d(k))) \leq \epsilon$. Then $\sum_{i \in N - \{d\}} |g_i(p(k'))| \leq \epsilon$, for all $k' \geq k$.

Proof: It follows from the proof of Proposition 3.4 that $g_i(p(k)) \leq 0$, for all k and for all $i \in N - \{d\}$. Moreover it follows from (2.6) that

$$g_d(p(k)) = -\sum_{i \in N - \{d\}} g_i(p(k)), \text{ for all } k. \tag{4.2}$$

By the monotonically nondecreasing property of ∇c^*_{ij} it follows also from (2.6) that g_d is monotonically nonincreasing in the j-th coordinate for all $j \in N - \{d\}$. Therefore, it follows from (3.8) and (4.1) that

$$g_d(p(k)) \leq g_d(p_1(\tau_1^d(k)), ..., p_n(\tau_n^d(k))), \tag{4.3}$$

and the result follows from (4.2), (4.3), and Proposition 4.2.

Q.E.D.

A similar result can analogously be shown for asynchronous iterative algorithms with flexible communication associated with an m-continuous supermapping $\check{F}$ on P" starting from $p(0) \in P$" such that $p(0) \geq \overline{F}(p(0))$.

For the nonlinear network flow problem studied in this paper the detection of the global termination of asynchronous iterative algorithms with flexible communication is achieved by using a local condition. A similar termination test is used for asynchronous iterations. We note that the termination test is $g_d(p(k)) < \epsilon$ in the synchronous case. Finally, we note that the termination test can be computed in parallel with price updating.

5. COMPUTATIONAL RESULTS

5.1. Problems and algorithms

We have considered hydraulic network flow problems with the following cost functions:

$$c_{ij}(f_{ij}) = |f_{ij}|^{\frac{1+b}{b}} .$$

This cost function satisfies Assumptions 2.1 to 2.3 and Assumption 3.2 on a bounded subdomain. We have taken $b = 1.85$; this case corresponds to turbulent flow in pipes (see [Bid65], see also [Por69] and [Rhe70]). We have:

$$\nabla c_{ij}^*(p_i - p_j) = sign(p_i - p_j)\, |p_i - p_j|^{1.85} .$$

The network topology always corresponds to a grid-like network with low degree nodes ($\max_{i \in N} a_i = 4$). The number of nodes and arcs varies from 48 to 144 and 77 to 237, respectively. For all problems there are three nonzero traffic inputs which are equal to $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$, respectively and three nonzero traffic outputs which are also equal to $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$, respectively.

We have considered various sequential iterative methods associated with submappings: the relaxation method with inexact line search (I) proposed in [ChZ91 p. 876] (see also subsection 3.3.1), a gradient method (G) (see subsection 3.3.2), and the gradient type method presented in Remark 3.2. The latter method is considered with different accuracies of the test $\left| g_i(p_i^{q'}; p) \right| \leq \epsilon' : \epsilon' = 10^{-3}$ and $\epsilon' = 10^{-2}$, respectively; in the sequel these versions are denoted by T and t, respectively. The stepsize of the gradient and gradient type methods is 0.34. All algorithms terminate when the deficit at node d is less than 0.1.

Parallel synchronous and asynchronous implementations of I, G, T, and t and the asynchronous implementation with flexible communication of the algorithm t presented in subsection 4.3 were carried out on the distributed memory architecture; they are denoted by SIx, AIx, SGx, AGx, STx, ATx, Stx, Atx, and Ftx, respectively, where the first letter (i.e. S, A, or F) corresponds to the type of implementation (which can be synchronous, asynchronous, or asynchronous with flexible communication, respectively) and x is the number of processors which is equal to 2, 4, 8, or 16. All parallel algorithms terminate when the deficit at node d is less than 0.1 (see subsection 4.3.1). Clearly, it follows from (4.2) that the absolute value of the deficit at each node is small compared with 0.1. In the case of asynchronous gradient type algorithms with flexible communication, the current value of the components of the iteration vector is communicated when $q = 3$. For a given problem, the starting point is always the same and satisfies $p_i \leq \underline{F}_i(p)$ for all $i \in N - \{d\}$. For all problems and algorithms we balance the number of nodes assigned to the different processors. Task scheduling is made according to static mode.

5.2. Experimental results

Figure 2 displays the solution times in seconds of I, G, T, and t in function of the number of nodes in the network. Table 1 gives the solution times in seconds of the different parallel iterative methods for a problem with 144 nodes and 234 arcs solved by using 16 processors. The speedups of the parallel iterative methods are reported in Tables 2 to 10 for different problems. The column size gives the number of nodes in the network.

Algorithm	SI16	AI16	SG16	AG16	ST16	AT16	St16	At16	Ft16
time (s)	111.33	107.76	66.35	66.67	100.53	84.99	55.39	52.74	48.59

Table 1 : times of the parallel iterative methods

Size \ Algorithm	SI2	SI4	SI8	SI16
48	1.92	3.48	6.24	-
72	1.95	3.64	6.76	-
96	1.96	3.73	7.04	12.68
144	1.97	3.81	7.33	13.64

Table 2 : speedups of synchronous relaxation algorithms with inexact line search

Size \ Algorithm	AI2	AI4	AI8	AI16
48	1.96	3.68	6.17	-
72	1.98	3.85	7.00	-
96	1.99	3.91	7.34	12.45
144	1.99	3.95	7.68	14.09

Table 3 : speedups of asynchronous relaxation algorithms with inexact line search

Size \ Algorithm	SG2	SG4	SG8	SG16
48	1.96	3.84	7.56	-
72	1.97	3.89	7.69	-
96	1.98	3.91	7.76	15.29
144	1.98	3.94	7.83	15.49

Table 4 : speedups of synchronous gradient algorithms

Algorithm / Size	AG2	AG4	AG8	AG16
48	1.99	3.92	7.51	-
72	1.99	3.95	7.70	-
96	1.99	3.96	7.80	15.04
144	1.99	3.97	7.87	15.42

Table 5 : speedups of asynchronous gradient algorithms

Size \ Algorithm	ST2	ST4	ST8	ST16
48	1.82	2.97	5.08	-
72	1.84	3.11	5.46	-
96	1.84	3.19	5.82	10.16
144	1.83	3.26	6.11	11.03

Table 6 : speedups of synchronous gradient type algorithms ($\epsilon' = 10^{-3}$)

Size \ Algorithm	AT2	AT4	AT8	AT16
48	1.97	3.66	5.68	-
72	1.97	3.81	6.58	-
96	1.96	3.82	7.05	11.24
144	1.94	3.80	7.34	12.97

Table 7 : speedups of asynchronous gradient type algorithms ($\epsilon' = 10^{-3}$)

Size \ Algorithm	St2	St4	St8	St16
48	1.76	3.09	5.39	-
72	1.76	3.23	5.87	-
96	1.79	3.35	6.20	10.83
144	1.83	3.50	6.64	12.18

Table 8 : speedups of synchronous gradient type algorithms ($\epsilon' = 10^{-2}$)

Size \ Algorithm	At2	At4	At8	At16
48	1.83	3.36	5.5	-
72	1.82	3.48	6.25	-
96	1.83	3.54	6.60	10.97
144	1.85	3.64	7.05	12.80

Table 9 : speedups of asynchronous gradient type algorithms ($\epsilon' = 10^{-2}$)

Size \ Algorithm	Ft2	Ft4	Ft8	Ft16
48	1.85	3.54	6.6	-
72	1.85	3.57	6.75	-
96	1.86	3.61	6.94	13.10
144	1.88	3.71	7.27	13.89

Table 10 : speedups of asynchronous gradient type algorithms with flexible communication ($\epsilon' = 10^{-2}$)

We have also assigned an unbalanced number of nodes to the different processors for a problem with 120 nodes and 197 arcs solved by using 16 processors, the times in seconds of the different parallel iterative methods are reported in Table 11.

Algorithm	SI16	AI16	SG16	AG16	ST16	AT16	St16	At16	Ft16
time (s)	80.03	67.94	46.62	40.95	75.32	56.44	41.32	34.39	31.24

Table 11 : unbalanced case, times of the parallel iterative methods

5.3. Analysis of the computational experience

Figure 2 shows that t was faster than T. Moreover t and G were faster than I. We note in particular that I requires more computation at each iteration than G.

There was deterministic load balancing in the particular case of parallel gradient algorithms since processors computed essentially a gradient at each updating and very regular network topologies were considered, i.e. grid-like networks that can be partitioned and assigned equitably to the different processors. Table 5 shows that an asynchronous implementation has speeded up very efficiently the gradient method. Asynchronous gradient algorithms were generally faster than synchronous gradient algorithms for a sufficiently large granularity. The very good performances of synchronous gradient algorithms reported in Table 4 resulted from the regular network topologies considered, accordingly idle times due to synchronization were very small. From Table 1 it can be seen that the very good speedups of synchronous and asynchronous gradient algorithms were not sufficient to render these parallel algorithms faster than all other methods. We recall that G was slower than t.

There was also deterministic load balancing in the case of parallel relaxation algorithms with inexact line search since each iteration consisted of a single step, processors computed essentially gradients at each updating, and very regular network topologies were considered. Table 3 shows that an asynchronous implementation has speeded up very efficiently the relaxation method with inexact line search. Asynchronous implementations were faster than synchronous implementations for a sufficiently large granularity. The good performances of synchronous algorithms illustrated in Table 2 were also due to the fact that we have considered essentially very regular network topologies.

Parallel gradient type algorithms led to nondeterministic load unbalancing since the gradient type mapping results from an iterative process. Tables 7 and 9 show that an asynchronous implementation has speeded up efficiently the gradient type method. From Tables 1 and 6 to 9 it can be seen that synchronous gradient type methods were slower than asynchronous gradient type methods. Idle time due to synchronization was great in this case since there was nondeterministic load unbalancing. Tables 1, 7, and 9 also show that the speedups of the asynchronous gradient type methods increased with the accuracy of the line search test. However, the gain of efficiency was not sufficient to render AT faster than At, since T presented poor performances.

Asynchronous gradient type methods with flexible communication were faster than all other methods. In particular St, At, and Ft differed only by the implementation which was synchronous, asynchronous, and asynchronous with flexible communication, respectively. From Tables 1 and 8 to 10 it can be seen that an asynchronous implementation with flexible communication of the gradient type method was more efficient than a synchronous or an asynchronous implementation.

In the unbalanced case, idle time due to synchronization were greater. From Table 11 it can be seen that the asynchronous gradient type method with flexible communication (Ft) was faster than all other methods in this case too.

6. CONCLUSIONS

In this paper, we have studied the solution of the dual of a strictly convex network flow problem via a new class of parallel asynchronous iterative methods: parallel asynchronous iterations with flexible communication. In this new class of methods, the current value of the components of the iteration vector resulting from intermediary steps of updating can be communicated to other processors. We have shown the monotone convergence of asynchronous iterative algorithms with flexible communication. A preliminary computational experience using a distributed memory architecture has mainly shown that an asynchronous implementation with flexible communication is more efficient than a synchronous or totally asynchronous implementation.

7. APPENDIX : PROOF OF PROPOSITION 3.4

A.) We first show by induction that the sequence $\{p(k)\}$ given by (3.6) to (3.11) is well defined and satisfy:

$$p_i(\tau_i^i(k)) \leq \underline{F}_i(p_1(\tau_1^i(k)),...,p_n(\tau_n^i(k))) \text{ for all } k \in T^i \text{ and } i \in N-\{d\},$$

$$p_j(\tau_j^i(k)) \leq \underline{p}_j, \text{ for all } k \in T^i, j \in N-\{d\}, \text{ and } i \in N-\{d\},$$

$$p(0) \leq ... \leq p(k) \leq p(k+1) \leq ... \leq \underline{p}, k=0,1,...$$

$$p(k) \leq \underline{F}(p(k)), k=0,1,...$$

$$g_i(p(k)) \leq 0, \text{ for all } i \in N-\{d\}, k=0,1,...$$

From (3.8), (3.9), and the definition of $p(0)$ we have:

$$p_1(\tau_1^i(1)),...,p_n(\tau_n^i(1)) = p(0) \leq \underline{F}(p(0)) = \underline{F}(p_1(\tau_1^i(1)),...,p_n(\tau_n^i(1))) \text{ and } p(0) \leq \underline{p}. \tag{7.1}$$

Consider i such that $1 \in T^i$, it follows from (7.1), (3.6), Definition 3.1, and Remark 3.1 that we have

$$p_i(0) = p_i(\tau_i^i(1)) \leq p_i(1) \leq \underline{p}_i. \tag{7.2}$$

Consider now the case where $1 \notin T^i$, clearly it follows from (3.6) and the definition of $p(0)$ that

$$p_i(1) = p_i(0) \leq \underline{p}_i. \tag{7.3}$$

Thus (7.2) and (7.3) imply

$$p(0) \leq p(1) \leq \underline{p}. \tag{7.4}$$

By the monotonicity of $\underline{F}$, it follows from (7.4) that

$$\underline{F}_i(p(0)) \leq \underline{F}_i(p(1)), \text{ for all } i \in N - \{d\}. \tag{7.5}$$

If $1 \in T^i$, then it follows from (3.6), (7.1), Definition 3.1, and (7.5) that

$$p_i(1) \leq \check{F}_i(p(0)) \leq \underline{F}_i(p(0)) \leq \underline{F}_i(p(1)). \tag{7.6}$$

If $1 \notin T^i$, then it follows from (3.6), (7.1), and (7.5) that

$$p_i(1) = p_i(0) \leq \underline{F}_i(p(0)) \leq \underline{F}_i(p(1)). \tag{7.7}$$

Thus it follows from (7.6) and (7.7) that

$$p(1) \leq \underline{F}(p(1)). \tag{7.8}$$

By the monotonically nondecreasing property of g_i in the i-th coordinate it follows from (7.8) and (3.1) that

$$g_i(p(1)) \leq g_i(\underline{F}_i(p(1)); p(1)) = 0, \text{ for all } i \in N - \{d\}. \tag{7.9}$$

The inequalities (7.1), (7.4), (7.8), and (7.9) give the first step of the induction.

Now, assume that for a given $k, k \geq 1$, we have for all m satisfying $0 \leq m \leq k-1$:

$$p_i(\tau_i^i(m)) \leq \underline{F}_i(p_1(\tau_1^i(m)), ..., p_n(\tau_n^i(m))) \text{ for all } m \in T^i, i \in N - \{d\}, \tag{7.10}$$

$$(p_1(\tau_1^i(m)), ..., p_n(\tau_n^i(m))) \leq \underline{p}, \text{ for all } m \in T^i \text{ and } i \in N - \{d\},$$

$$p(0) \leq ... \leq p(m-1) \leq p(m) \leq ... \leq p(k-1) \leq \underline{p}, \tag{7.11}$$

$$p(m) \leq \underline{F}(p(m)), \tag{7.12}$$

$$g_i(p(m)) \leq 0, \text{ for all } i \in N - \{d\}. \tag{7.13}$$

It follows from (3.9) that

$$p_i(\tau_i^i(k)) = p_i(k-1), \text{ for all } k \in T^i. \tag{7.14}$$

From (3.8) to (3.10), and (7.11) it follows that we have

$$p(0) \le (p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \le \underline{p}. \tag{7.15}$$

We introduce the following notation:

$$T^i(k) = \{t \in T^i | 0 \le t < k\},$$

where $i \in N - \{d\}$ and $k \in \mathcal{N}$, the set of natural numbers.

If $k \in T^i$ and $T^i(k) = \emptyset$, then it follows from (3.6) that

$$p_i(k-1) = p_i(0). \tag{7.16}$$

By the monotonicity of $\underline{F}$ it follows from (7.14), (7.16), (7.1), and (7.15) that

$$p_i(\tau_i^i(k)) = p_i(k-1) = p_i(0) \le \underline{F}_i(p(0)) \le \underline{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \text{ and } (p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \le \underline{p}. \tag{7.17}$$

If $k \in T^i$ and $T^i(k) \ne \emptyset$, then it follows from (3.6) that we have

$$p_i(k-1) = p_i(l), \tag{7.18}$$

where $l = \max_{t \in T^i(k)} t$. It follows from (7.14) and (7.18) that we have

$$p_i(\tau_i^i(k)) = p_i(k-1) = p_i(l). \tag{7.19}$$

Moreover it follows from (3.8) to (3.10) and (7.11) that we have

$$(p_1(\tau_1^i(l)), ..., p_n(\tau_n^i(l))) \le (p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \le \underline{p}. \tag{7.20}$$

By the monotonicity of $\underline{F}$ and Definition 3.1, it follows from (7.19), (3.6), and (7.20) that

$$p_i(\tau_i^i(k)) = p_i(l) \le \check{F}_i(p_1(\tau_1^i(l)), ..., p_n(\tau_n^i(l))) \le \underline{F}_i(p_1(\tau_1^i(l)), ..., p_n(\tau_n^i(l))) \le \tag{7.21}$$

$$\underline{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \text{ and } (p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \le \underline{p}.$$

It follows from (7.17) and (7.21) that

$$p_i(\tau_i^i(k)) \le \underline{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \text{ and } (p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \le \underline{p}, k \in T^i. \tag{7.22}$$

The inequalities (7.22) extend the inequalities (7.10) to rank k.

The relations (7.14), (3.6), (7.22), Definition 3.1, and Remark 3.1 imply that if $k \in T^i$, then we have

$$p_i(k-1) = p_i(\tau_i^i(k)) \le p_i(k) \le \underline{p}_i. \tag{7.23}$$

Therefore it follows from (7.23), (3.6), and (7.11) that

$$p(k-1) \leq p(k) \leq \underline{p}. \tag{7.24}$$

We note that (7.24) extends (7.11) to rank k.

By the monotonicity of $\underline{F}$ and Definition 3.1, if $k \in T^i$, then it follows from (3.6), (3.8) to (3.10), (7.11), and (7.24) that

$$p_i(k) \leq \check{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \leq \underline{F}_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \leq \underline{F}_i(p(k)). \tag{7.25}$$

Moreover, by the monotonicity of $\underline{F}$, if $k \notin T^i$ and $T^i(k) = \emptyset$, then it follows from (3.6), (7.1), (7.11), and (7.24) that

$$p_i(k) = p_i(k-1) = p_i(0) \leq \underline{F}_i(p(0)) \leq \underline{F}_i(p(k)), \tag{7.26}$$

finally, if $k \notin T^i$ and $T^i(k) \neq \emptyset$, then it follows from (3.6), (7.12), (7.11), and (7.24) that

$$p_i(k) = p_i(k-1) = p_i(l) \leq \underline{F}_i(p(l)) \leq \underline{F}_i(p(k)), \tag{7.27}$$

where $l = \max_{t \in T^i(k)} t$, So, (7.25) to (7.27) imply

$$p(k) \leq \underline{F}(p(k)). \tag{7.28}$$

The inequality (7.28) extends (7.12) to rank k.

By the monotonically nondecreasing property of g_i in the i-th coordinate, it follows from (7.28) and (3.1) that

$$g_i(p(k)) \leq g_i(\underline{F}_i(p(k)); p(k)) = 0, \text{ for all } i \in N - \{d\}. \tag{7.29}$$

The inequality (7.29) extends (7.13) to rank k. Thus the induction is complete.

B.) The monotonically increasing sequence $\{p(k)\}$ is upperbounded by $\underline{p}$. Hence $\{p(k)\}$ is convergent and there exists $\check{p} \leq \underline{p}$ such that

$$p(k) \uparrow \check{p}, k \to \infty. \tag{7.30}$$

Moreover by the continuity of $\underline{F}$ (see [BeE87]), it follows from (7.30) and (7.28) that $\underline{F}(p(k)) \uparrow \underline{F}(\check{p}), k \to \infty$, and $\check{p} \leq \underline{F}(\check{p})$.

C.) Now we show that the sequence $\{p(k)\}$ defined by (3.6) to (3.11) satisfies $p(k) \uparrow \underline{p}, k \to \infty$. It follows from (3.6) that for all $k \geq 1$ such that $k \in T^i$ and $T^i(k) \neq \emptyset$, we have

$$p_i(k-1) = p_i(l), \tag{7.31}$$

where

$$l = \max_{t \in T^i(k)} t. \tag{7.32}$$

We introduce the sequence $\{z(m)\}$ such that for all $i \in N - \{d\}$

$$z_i(2m) = p_i(k), m = card(T^i(k)), k \in T^i, \tag{7.33}$$

and

$$z_i(2m-1) = \check{F}'_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))), m = card(T^i(k)), k \in T^i, m \geq 1, \tag{7.34}$$

where $\check{F}'$ is the order continuous submapping on P' which minorizes $\check{F}$. We note that

$$\lim_{k \to \infty} card(T^i(k)) = +\infty$$

since T^i is infinite. Clearly it follows from (7.30) and (7.33) that for all $i \in N - \{d\}$ we have

$$z_i(2m) \uparrow \check{p}_i, m \to \infty. \tag{7.35}$$

It follows from (7.34), (3.3), (7.14), (3.6), (7.33), (7.31), and (7.32) that for all $i \in N - \{d\}$ and $m \geq 1$ we have

$$z_i(2m-1) = \check{F}'_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \in [p_i(k-1), p_i(k)] = [z_i(2m-2), z_i(2m)]. \tag{7.36}$$

From (7.35) and (7.36), for all $i \in N - \{d\}$ we have

$$z_i(2m-1) \uparrow \check{p}_i, m \to \infty. \tag{7.37}$$

If $m = card(T^i(k)) \to \infty$, then $k \to \infty$. It follows from (7.34) and (7.37) that for all $i \in N - \{d\}$ we have

$$\check{F}'_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \uparrow \check{p}_i, k \to \infty. \tag{7.38}$$

From (3.11), and (7.30), for all $i \in N - \{d\}$ we have

$$(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \uparrow \check{p}_i, k \to \infty. \tag{7.39}$$

By the order continuity of $\check{F}'$, it follows from (7.39) that for all $i \in N - \{d\}$ we have

$$\check{F}'_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))) \uparrow \check{F}'_i(\check{p}), k \to \infty. \tag{7.40}$$

Therefore it follows from (7.38) and (7.40) that we have $\check{F}'_i(\check{p}) = \check{p}_i$, for all $i \in N - \{d\}$. From Definition 3.1, $\check{F}'_i(\check{p}) = \check{p}_i$ for all $i \in N - \{d\}$ implies $\underline{F}_i(\check{p}) = \check{p}_i$ for all $i \in N - \{d\}$, where $\underline{F}$ is the minimal relaxation mapping defined by (3.1). Thus, it follows from the definition of $\underline{p}$ that $\check{p} = \underline{p}$.

Q.E.D.

Acknowledgments: The authors wish to thank the reviewers for their helpful remarks and comments.

References

[Bau78] G. M. Baudet, *Asynchronous iterative methods for multiprocessors*, J. Assoc. Comput. Mach., 2 (1978), pp. 226-244.

[Ber82] D. P. Bertsekas, *Distributed dynamic programming*, IEEE Trans. Auto. Contr., AC-27 (1982), pp. 610-616.

[Ber83] D. P. Bertsekas, *Distributed asynchronous computation of fixed points*, Mathematical Programming, 27 (1983), pp. 107-120.

[BeC90] D. P. Bertsekas and D. A. Castañon, *Parallel asynchronous primal-dual methods for the minimum cost flow problem*, report LIDS-P-1998, Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA, 1990.

[BeC91] D. P. Bertsekas and D. A. Castañon, *Parallel synchronous and asynchronous implementation of the auction algorithm*, Parallel Computing, 17 (1991), pp. 707-732.

[BCE95] D. P. Bertsekas D. A. Castañon, J. Eckstein, and S. Zenios, *Parallel computing in optimization*, in M. O. Bal et al. Eds., Handbooks in Operation Research and Management Science, 7 (1995), Elsevier Science, NY, pp.331-339.

[BeE87] D. P. Bertsekas and D. El Baz, *Distributed asynchronous relaxation methods for convex network flow problems*, SIAM J. on Control and Optimization, 25 (1987), pp. 74-85.

[BeT89] D. P. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation, Numerical Methods*, Prentice Hall, Englewood Cliffs, N.J., 1989.

[BHT87] D. P. Bertsekas, P. Hossein, and P. Tseng, *Relaxation methods for network flow problems with convex arc costs*, SIAM J. on Control and Optimization, 25 (1987), pp. 1219-1243.

[BiD65] G. Birkhoff and J.B. Diaz, *Nonlinear network problems*, Quart. Appl. Math., 13 (1965), pp. 431-443.

[ChM69] D. Chazan and W. Miranker, *Chaotic relaxation,* Linear Algebra Appl., 2 (1969), pp. 199-222.

[ChZ91] E. Chajakis and S.A. Zenios, *Synchronous and asynchronous implementations of relaxation algorithms* *for nonlinear network optimization*, Parallel Computing, 17 (1991) pp. 873-894.

[Elb90] D. El Baz, *M-functions and parallel asynchronous algorithms,* SIAM Journal on Numerical Analysis, 27 (1990), pp. 136-140.

[Elb91] D. El Baz, *Asynchronous iterative algorithms for convex network flow problems*, Proceedings of the European Control Conference, Grenoble, France (1991), pp. 2397-2402.

[Elb93] D. El Baz, *Asynchronous implementation of relaxation and gradient algorithms for convex network flow problems*, Parallel Computing, 19 (1993), pp. 1019-1028.

[Elb94a] D. El Baz, *Nonlinear systems of equations and parallel asynchronous iterative algorithms*, in Advances in Parallel Computing, vol. 9, Parallel Computing Trends and Applications, Joubert et al. editors, North Holland, Amsterdam, 1994, pp. 89-96.

[Elb94b] D. El Baz, *Parallel iterative algorithms for the solution of Markov systems*, in the Proceedings of the 33rd IEEE Conference on Decision and Control, Lake Buena Vista, Florida, December 1994, pp. 2524 2527.

[Elb96] D. El Baz, *Asynchronous gradient algorithms for a class of convex separable network flow problems*, Computational Optimization and Applications, 5 (1996), pp. 187-205.

[Elt82] M. N. El Tarazi, *Some convergence results for asynchronous algorithms*, Numerisch Mathematik, 39 (1982), pp. 325-340.

[Elt84] M. N. El Tarazi, *Algorithmes mixtes asynchrones, etude de la convergence monotone*, Numerisch Mathematik, 44 (1984), pp. 363-369.

[GiS91] L. Giraud, P. Spiteri, *Résolution parallèle de problèmes aux limites non linéaires*, MMAN, 25 (1991), pp. 579-606.

[LB87] S. Li and T. Basar, *Asymptotic agreement and convergence of asynchronous stochastic algorithms*, IEEE Trans. Auto. Contr., AC-32 (1987), pp. 612-618.

[Mie75a] J. C. Miellou, *Algorithmes de relaxation chaotique à retards*, RAIRO, R1 (1975), pp. 55-82.

[Mie75b] J. C. Miellou, *Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés*, C.R.A.S. Paris, 280 (1975), pp. 233-236.

[Mie86] J. C. Miellou, *Asynchronous iterations and order intervals*, in Parallel Algorithms and Architectures, M. Cosnard ed., North Holland (1986), pp. 85-96.

[MCB90] J. C. Miellou, Ph. Cortey-Dumont, and M. Boulbrachêne, *Perturbation of fixed point iterative methods*, Advances in Parallel Computing, 1 (1990), pp. 81-122.

[MES94] J.C. Miellou, D. El Baz, P. Spiteri, *A new class of asynchronous iterative algorithms with order intervals*, LCS/LAAS/IRIT Report 1994-16 (1994).

[MiS85a] J. C. Miellou, P. Spiteri, *Two criteria for the convergence of asynchronous iterations*, in Computers and Computing (P. Chenin, C. di Crescenzo, and F. Robert eds), Wiley- Masson, Paris (1985), pp. 91-95.

[MiS85b] J. C. Miellou, P. Spiteri, *Un critère de convergence pour des méthodes générales de point fixe*, R.A.I.R.O. MMAN, 19 (1985), pp. 645-669.

[OrR70] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.

[Por69] T. A. Porshing, *Jacobi and Gauss-Seidel methods for nonlinear network problems*, SIAM J. Numer. Anal., 6 (1969), pp. 437-449.

[Rhe70] W. C. Rheinboldt, *On M-functions and their application to nonlinear Gauss-Seidel iterations and to network flows,* J. Mathematical Analysis and Applications, 32 (1970), pp. 274-307.

[Roc70] R. T. Rockafellar, *Convex Analysis,* Princeton University Press, Princeton New Jersey, 1970.

[Roc84] R.T. Rockafellar, *Network Flows and Monotropic Optimization,* John Wiley & Sons, New York, 1984.

[SME95] P. Spiteri, J.C. Miellou, and D. El Baz, *Asynchronous alternating Schwarz method for the solution of nonlinear partial differential equations*, IRIT/LCS/LAAS report 1994.

[TsB86] J. N. Tsitsiklis and D. P. Bertsekas, *Distributed asynchronous optimal routing in data networks*, IEEE Trans. Auto. Contr., AC-31 (1986), pp. 325-332.

[TsB87] P. Tseng and D. P. Bertsekas, *Relaxation methods for problems with strictly convex separable costs and linear constraints*, Math. Prog., 38 (1987), pp. 303-321.

[TBA86] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms*, IEEE Trans. Auto. Contr., AC-31 (1986), pp. 803-812.

[TBT90] P. Tseng, D. P. Bertsekas, and J. N. Tsitsiklis, *Partially asynchronous parallel algorithms for network flow and other problems,* SIAM J. Control and Optimization, 28 (1990) pp. 678-710.

[Tse90] P. Tseng, *Dual coordinate ascent for problems with strictly convex costs and linear constraints, a unified approach*, SIAM J. Contr. and Optimization, 28 (1990), pp. 214-242.

[UrD90] A. Uresin and M. Dubois, *Sufficient conditions for the convergence of asynchronous iterations*, Parallel

Computing, 10 (1989), pp. 83-92.

[ZeL88] S. Zenios and R. Lasken, *The Connection Machines CM-1 and CM-2: solving nonlinear network problems,* International Conference on Supercomputing, St Malo, France, (1988), pp. 648-658.

[ZeM88] S. Zenios and J. Mulvey, *A distributed algorithm for convex network optimization problems*, Parallel Computing, 6 (1988), 45-56.

Figure 1:

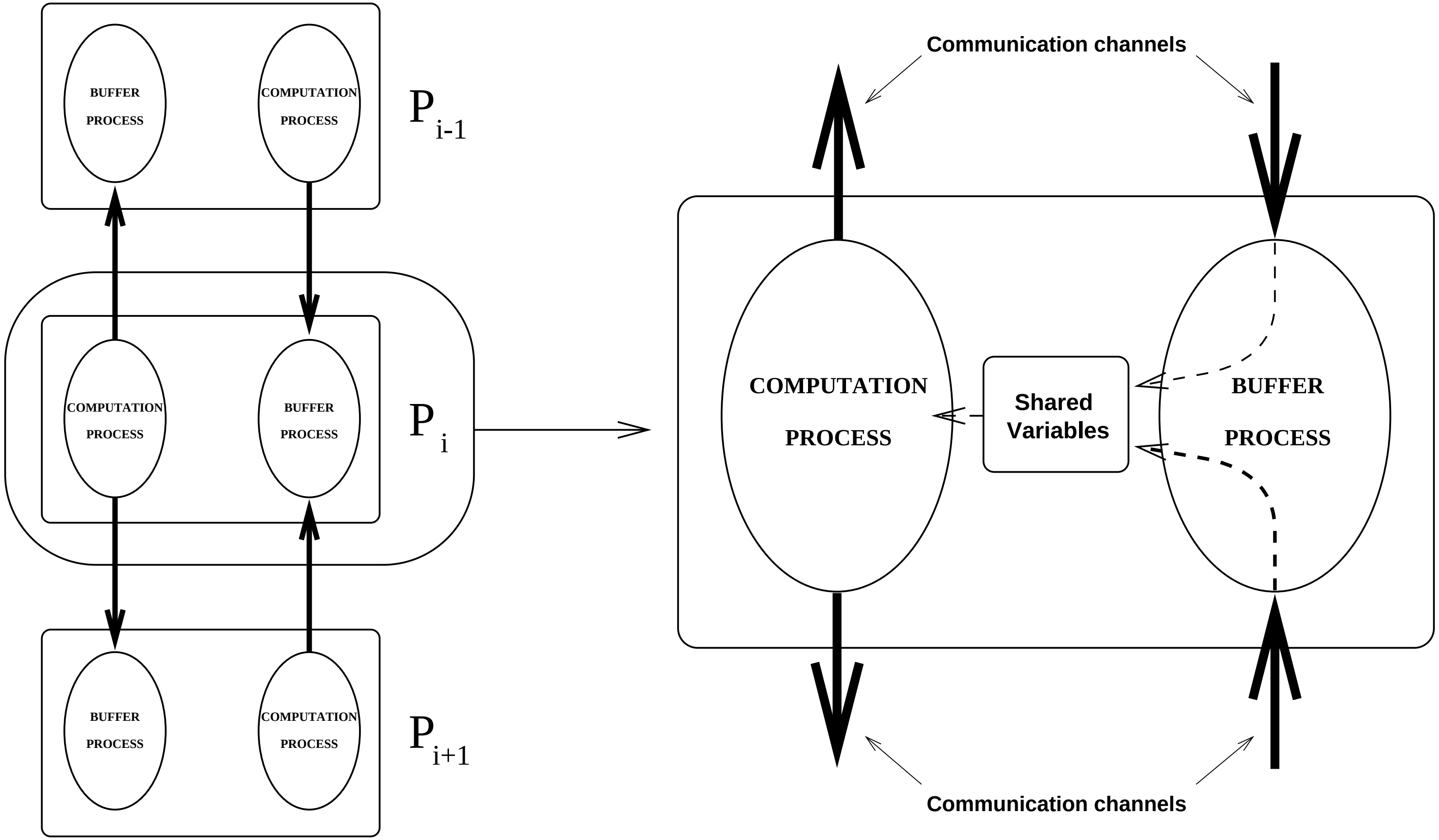

Figure 1: Implementation with buffer of asynchronous iterations.

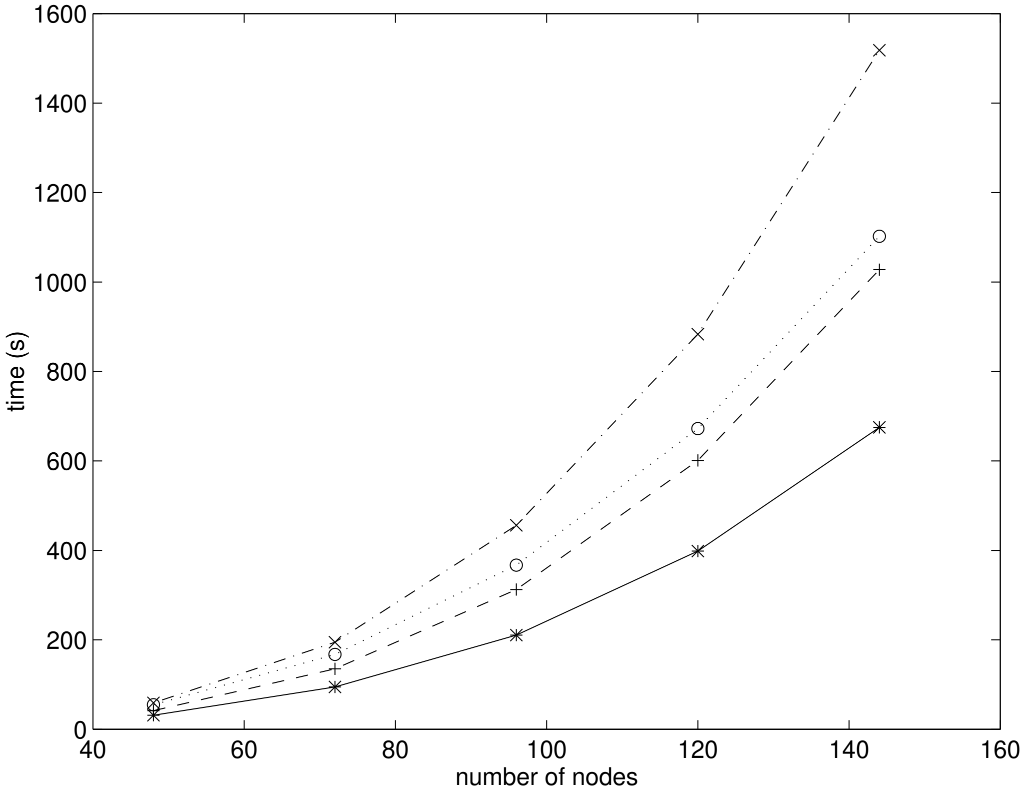

Figure 2: time of I (dashdot), G (dashed), T (dotted), and t (solid).