ASYNCHRONOUS GRADIENT ALGORITHMS

FOR A CLASS OF CONVEX SEPARABLE

NETWORK FLOW PROBLEMS

Didier El Baz

LAAS du CNRS

7, avenue du Colonel Roche

31077 Toulouse Cedex France

ABSTRACT

We consider the single commodity strictly convex network flow problem. The dual of this problem is unconstrained, differentiable, and well suited for solution via distributed or parallel iterative methods. We present and prove convergence of gradient and asynchronous gradient algorithms for solving the dual problem. Computational results are given and analysed.

Key words. network flow problems, distributed algorithms, parallel computation, asynchronous iterative methods, gradient methods.

1. INTRODUCTION

Asynchronous iterative methods whereby iterations are carried out in parallel by several processors in arbitrary order and without any synchronization have been devised for parallel and distributed computing systems. The restrictions imposed on asynchronous iterative methods are very weak: no component of the iterate vector is abandoned forever and more and more recent values of the components have to be used as the computation progresses. Asynchronous iterative methods have been implemented very early in distributed systems. We note for example the routing algorithm originally implemented in the Arpanet in 1969 (see [3]). There is now a great deal of computational experience available with asynchronous iterative algorithms on parallel computers (see [1], [2], [5]- [7], [13], [15], [18], and [32]). The advantages of asynchronous iterative algorithms are implementation simplicity, computation flexibility, and potential tolerance to problem data changes. Since there is no synchronization overhead, one may also hope that asynchronous iterations will converge faster. Therefore it is desirable to know the conditions which ensure the convergence of asynchronous iterative algorithms. There are many results in the literature relevant to the convergence of parallel or distributed asynchronous iterative methods. For linear systems of equations $Fx = z$, the necessary and sufficient condition for the convergence of asynchronous relaxation algorithms is: the matrix F is a so-called H-matrix (see [14]). For other convergence results, reference is made to [16] and [23]. For nonlinear systems of equations, convergence results for asynchronous relaxation algorithms have been given when F is off-diagonally antitone and diagonally isotone (see [17] and [19]), an M-function (see [17], [20], and [26]), or H-accretive (see [27]). The reader is also referred to [28]. For nonlinear fixed point problems, the convergence of asynchronous relaxation methods was established for P-contraction mappings (see [2], [4], [9], and [24]), contraction mappings (see [21]), and isotone and continuous fixed point mappings (see [3], [4], [8], [9], [25], and [26]), see also [22]. Asynchronous gradient algorithms for unconstrained optimization are guaranteed to converge if the Hessian matrix of the cost function satisfies a diagonal dominance condition (see [4] and [9, Section 6.3]). More generally all these results can be considered as examples of application of the Asynchronous Convergence Theorem of Bertsekas and Tsitsiklis (see [9, p.431], see also [4, p.114]). This theorem is also a powerful aid in showing convergence of asynchronous iterative algorithms, as we shall see in Section 3.

This paper deals with the single commodity strictly convex network flow problem. We concentrate on the dual of this problem which is unconstrained, differentiable, and well suited for solution via distributed or parallel iterative methods. In [8], we have shown that the structure of the dual problem allows the successful application of asynchronous relaxation methods. In this paper we present and prove convergence of gradient and asynchronous gradient methods for solving the dual problem. The convergence results are all new. In particular, the convergence result for asynchronous gradient methods is really different from the convergence result of Bertsekas for unconstrained optimization, which is based upon the diagonal dominance of the Hessian matrix of the cost function (see [4] and [9, Section 6.3 and 7.5]). Computational results are also presented and analysed in this paper. Synchronous and asynchronous gradient and relaxation algorithms were carried out on a distributed memory multiprocessor.

Section 2 deals with the convex network flow problem. In Section 3 we prove convergence of gradient and asynchronous gradient methods. We present and analyse computational results in Section 4. Conclusions are given in Section 5.

2. THE CLASS OF CONVEX SEPARABLE NETWORK FLOW PROBLEMS

Let $G = (N, A)$ be a 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 d be the single destination node for network traffic, $b_i \geq 0$ the traffic input at node $i \in N - \{d\}$, and $b_d = -\sum_{i \in N-\{d\}} b_i$ the traffic output at d. The problem is to minimize total cost subject to a conservation of flow constraint at each node

$$min \sum_{(i,j)\in A} c_{ij}(f_{ij}), \tag{1}$$

subject to

$$\sum_{(i,j)\in A} f_{ij} - \sum_{(m,i)\in A} f_{mi} = b_i, \forall i \in N.$$

We assume that problem (1) has a feasible solution. We also make the following standing assumptions on c_{ij}:

(a) c_{ij} is strictly convex, and lower semicontinuous;

(b) the conjugate convex function of c_{ij}, defined by

$$c^*_{ij}(t_{ij}) = \sup_{f_{ij}}\{t_{ij}.f_{ij} - c_{ij}(f_{ij})\}, \tag{2}$$

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

(c) $0 = \mathit{arg\ min}_{f_{ij}} c_{ij}(f_{ij})$;

(d) there exists a constant $\beta \geq 0$, such that for all $(i,j) \in A$ and all $(\xi,\eta),(\xi',\eta') \in \Gamma_{ij}$ with $\xi' < \xi$, we have :

$$\eta - \eta' \geq \frac{1}{\beta}.(\xi - \xi'),$$

where $\Gamma_{ij} = \{(\xi,\eta) \in R^2 / c'_{ij-}(\xi) \leq \eta \leq c'_{ij+}(\xi)\}$ is the characteristic curve associated with c_{ij} (see [31, p. 320]).

Assumption (b) implies that $\lim_{|f_{ij}|\to+\infty} c_{ij}(f_{ij}) = +\infty$. Therefore the objective function of problem (1) has bounded level sets (see [30, Section 8]). It follows that there exists an optimal solution for problem (1) which must be unique in view of the strict convexity assumed in (a). 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) (see [30, pp. 218, 253]). Assumptions (c) and (d) will be discussed in the sequel. We note that assumptions (a), (b), (c), and (d) are naturally satisfied in many practical situations:

$c_{ij}(f_{ij}) = a_{ij}.|f_{ij}|+b_{ij}.f_{ij}^2$, with $a_{ij} \geq 0$ and $b_{ij} > 0$; $c_{ij}(f_{ij}) = a_{ij}.max\{f_{ij}^2, f_{ij}^4\}$, with $a_{ij} > 0$; $c_{ij}(f_{ij}) = (\frac{1}{a_{ij}-f_{ij}} + b_{ij}).f_{ij}$, if $0 \leq f_{ij} < a_{ij}$, and $c_{ij}(f_{ij}) = +\infty$, if $f_{ij} < 0$ or $a_{ij} < f_{ij}$, with $a_{ij} > 0$ and $b_{ij} \geq 0$.

Problem (1) is of great practical interest and has been studied for a long time. A dual problem for (1) is given by

$$\min_{p \in R^n} q(p), \tag{3}$$

subject to no constraints on the vector $p = \{p_i/i \in N\}$,

where q is the dual functional given by

$$q(p) = \sum_{(i,j)\in A} c_{ij}^*(p_i - p_j) - \sum_{i \in N} b_i.p_i. \tag{4}$$

We refer to p as a price vector and its components as prices. The ith price, p_i, is a Lagrange multiplier associated with the ith conservation of flow constraint. The duality between problems (1) and (3) is explored in great detail in [31]. The necessary and sufficient condition for optimality of a pair (f, p) is given in [30]. A feasible flow vector $f = \{f_{ij}/(i,j) \in A\}$ is optimal for (1) and a price vector $p = \{p_i/i \in N\}$ is optimal for (3) if and only if for all arcs $(i,j) \in A$

$$p_i - p_j \text{ is a subgradient of } c_{ij} \text{ at } f_{ij}.$$

An equivalent condition is

$$f_{ij} = \nabla c_{ij}^*(p_i - p_j), \forall (i,j) \in A.$$

Any one of these equivalent relations is referred to as the complementary slackness condition (see [30, pp. 337-338] and [8]).

Existence of an optimal solution of the dual problem can be guaranteed under the following additional regular feasibility assumption (see [31, p. 360 and p. 329]): there exists a feasible flow vector, $f = \{f_{ij}/(i,j) \in A\}$, such that $c'_{ij-}(f_{ij}) < +\infty$ and $c'_{ij+}(f_{ij}) > -\infty$, for all $(i,j) \in A$, where c'_{ij-}, respectively c'_{ij+}, denotes the left, respectively the right, derivative of c_{ij}. We note that the regular feasibility assumption is not overly restrictive. 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. We constrain the price of the destination node, p_d, to be zero. This choice will have important consequences in the following. We consider the reduced dual optimal solution set P^* defined by:

$$P^* = \{p'/q(p') = \min_p \; q(p), p'_d = 0\}. \tag{5}$$

Clearly P^* is nonempty.

From (4), it follows that

$$\left.\frac{\partial q}{\partial p_i}\right|_p = \sum_{(i,j)\in A} \nabla c^*_{ij}(p_i - p_j) - \sum_{(m,i)\in A} \nabla c^*_{mi}(p_m - p_i) - b_i. \tag{6}$$

From (2) and assumption (c), it follows that

$$\nabla c^*_{ij}(0) = 0. \tag{7}$$

From (6) and (7), it follows that $\left.\frac{\partial q}{\partial p_i}\right|_{\underline{p}} = -b_i \leq 0$ for all $i \in N - \{d\}$, where $\underline{p}$ denotes the vector of R^n with all components zero.

We recall that P^* is the nonempty set of vectors $p \in R^n$ such that $p_d = 0$ and for all arcs $(i,j) \in A$, $p_i - p_j$ is a subgradient of c_{ij} at f^*_{ij}, where $f^* = \{f^*_{ij}/(i,j) \in A\}$ is the unique primal optimal solution.

THEOREM 2.1. Let assumptions of Section 2 hold. The intersection, denoted by I, of P^* with the nonnegative orthant is nonempty.

Proof. We define first the sets A^* and N^* by

$$A^* = \{(i,j) \in A / f^*_{ij} \neq 0\}, \text{ and } N^* = \{i \in N - \{d\} / \exists (i,j) \in A^* \text{ or } \exists (m,i) \in A^*\}.$$

Suppose that there exists a vector $p \in P^*$ such that the set $N_p = \{i \in N^*/p_i < 0\}$ is nonempty, and consider the node of N_p with the smallest price value, this node will be denoted by i for simplicity. From the complementary slackness condition we have $\nabla c^*_{ij}(p_i - p_j) = 0$ if $(i,j) \notin A^*$, and $\nabla c^*_{mi}(p_m - p_i) = 0$ if $(m,i) \notin A^*$. From assumptions (a) and (c), and the definitions of prices p_i and p_d, it follows that $\nabla c^*_{ij}(p_i - p_j) < 0$ if $(i,j) \in A^*$, and $\nabla c^*_{mi}(p_m - p_i) > 0$ if $(m,i) \in A^*$. It follows from (6) that $\left.\frac{\partial q}{\partial p_i}\right|_p < 0$, which contradicts the fact that $p \in P^*$. Hence, if $p \in P^*$, then

$$p_i \geq 0, \forall i \in N^*. \tag{8}$$

From assumption (c), it follows that 0 is a subgradient of c_{ij} at 0. Hence, from (8) and the complementary slackness condition, it follows clearly that there exists at least one vector $p \in P^*$ such that $p_i \geq 0$ for all nodes $i \in N - N^*$.

Hence, the intersection, denoted by I, of P^* with the nonnegative orthant is nonempty.

Q.E.D.

Since I is nonempty and P^* is the set of vectors $p \in R^n$ such that $p_i - p_j$ is a subgradient of c_{ij} at f^*_{ij} for all $(i,j) \in A$, it follows that I is a nonempty polyhedral convex set which has a minimal element, denoted by p^*, i.e. for all $p \in I$, and all $i \in N$, $p^*_i \leq p_i$.

Throughout the paper the component-wise partial ordering on R^n will be also written as $p^* \leq p$.

We define the set P by $P = \{p \in R^n / \underline{p} \leq p \leq p^*\}$, where $\underline{p}$ is the vector of R^n with all components zero. Clearly P is closed and bounded.

In the following, the number of arcs incident to node i is denoted by a_i.

THEOREM 2.2. Under the hypotheses of Section 2, there exists a constant $\alpha = \beta . \max_{i \in N} a_i$ such that for all $p, p'' \in R^n$, with $p'' \leq p$, we have

$$\nabla q(p) - \nabla q(p'') \leq \alpha.(p - p''). \tag{9}$$

Proof. We recall that Γ_{ij} is the characteristic curve associated with c_{ij}. It can be shown that c^*_{ij} has Γ^{-1}_{ij} as its marginal cost curve: $\Gamma^{-1}_{ij} = \{(\eta, \xi) \in R^2 / \xi = \nabla c^*_{ij}(\eta)\}$ (see [31, p. 331]). Hence, from assumption (d), it follows clearly that there exists a constant $\beta > 0$, such that for all $(i,j) \in A$ and all $p, p' \in R^n$ with $p'_i \leq p_i$ and $p'_j = p_j$, we have

$$\nabla c^*_{ij}(p_i - p_j) - \nabla c^*_{ij}(p'_i - p'_j) \leq \beta.(p_i - p'_i).$$

Thus, for all $i \in N$ and all $p, p' \in R^n$ with $p'_i \leq p_i$ and $p'_j = p_j$ for $j \neq i$, we have

$$\left.\frac{\partial q}{\partial p_i}\right|_p - \left.\frac{\partial q}{\partial p_i}\right|_{p'} = \sum_{(i,j) \in A} (\nabla c^*_{ij}(p_i - p_j) - \nabla c^*_{ij}(p'_i - p'_j)) - \sum_{(m,i) \in A} (\nabla c^*_{mi}(p_m - p_i) - \nabla c^*_{mi}(p'_m - p'_i))$$

$$\leq \beta . a_i .(p_i - p'_i).$$

It follows that there exists a constant $\alpha = \beta . \max_{i \in N} a_i$ such that for all $i \in N$ and all $p, p' \in R^n$, with $p'_i \leq p_i$ and $p'_j = p_j$ for $j \neq i$, we have

$$\left.\frac{\partial q}{\partial p_i}\right|_{p} - \left.\frac{\partial q}{\partial p_i}\right|_{p'} \leq \alpha.(p_i - p_i').$$

By the convexity of c_{ij}^*, ∇c_{ij}^* is nondecreasing (i.e. whatever $t_{ij}, t_{ij}' \in R$, $t_{ij}' \leq t_{ij}$ implies $\nabla c_{ij}^*(t_{ij}') \leq \nabla c_{ij}^*(t_{ij})$). It follows from (6) that for all $i \in N$, and for all $p', p'' \in R^n$, with $p'' \leq p'$ and $p_i'' = p_i'$, we have

$$\left.\frac{\partial q}{\partial p_i}\right|_{p'} - \left.\frac{\partial q}{\partial p_i}\right|_{p''} \leq 0.$$

Thus, there exists a constant $\alpha = \beta.\max_{i \in N} a_i$ such that for all $i \in N$ and for all $p, p', p'' \in R^n$, with $p'' \leq p'$, $p_i'' = p_i' \leq p_i$ and $p_j' = p_j$ for $j \neq i$, we have

$$\left.\frac{\partial q}{\partial p_i}\right|_{p} - \left.\frac{\partial q}{\partial p_i}\right|_{p''} \leq \left.\frac{\partial q}{\partial p_i}\right|_{p} - \left.\frac{\partial q}{\partial p_i}\right|_{p'} \leq \alpha.(p_i - p_i') = \alpha.(p_i - p_i'').$$

Q.E.D.

3. GRADIENT AND ASYNCHRONOUS GRADIENT ALGORITHMS

We consider the solution of the dual problem via gradient algorithms. Reference is made to [8] - [13] and [29] for various iterative methods applied to equality- and interval-constrained problems. The gradient iteration is defined by

$$p(k+1) = p(k) - \frac{1}{\alpha}.\nabla q'(p(k)), k = 0, 1, ..., \tag{10}$$

where $\alpha = \beta.\max_{i \in N} a_i$ and $\nabla q'(p(k))$ is a vector of R^n with ith component equal to $\left.\frac{\partial q}{\partial p_i}\right|_{p(k)}$ for all $i \neq d$, or 0 for $i = d$.

We define the gradient mapping $F : R^n \to R^n$ by

$$F(p) = p - \frac{1}{\alpha}.\nabla q'(p). \tag{11}$$

From Theorem 2.2, it follows clearly that F is isotone on R^n (i.e. whatever $p, p' \in R^n$, $p' \leq p$ implies $F(p') \leq F(p)$). Since for all $(i,j) \in A$, c^*_{ij} is real valued and continuously differentiable, it follows from (6) that for all $i \in N$, $\frac{\partial q}{\partial p_i}$ is continuous on R^n. Hence, $\nabla q'$ and F are also continuous on R^n.

THEOREM 3.1. Let assumptions of Section 2 hold. The gradient iteration $\{\underline{p}(k)\}$ defined by (10) and starting from $\underline{p}(0) = \underline{p}$ (where $\underline{p}$ is the vector of R^n with all components zero, $\underline{p} \leq p^*$, and $\nabla q'(\underline{p}) \leq \underline{p}$) converges monotonically to p^*.

Proof. We proceed by induction. Consider $\underline{p}(0) = \underline{p}$, it follows from theorem 2.2 that

$$\nabla q'(p^*) - \nabla q'(\underline{p}(0)) \leq \alpha.(p^* - \underline{p}(0)),$$

since $\nabla q'(\underline{p}(0)) \leq \underline{p}$ and $\nabla q'(p^*) = \underline{p}$, we have

$$\underline{p}(0) \leq \underline{p}(1) = \underline{p}(0) - \frac{1}{\alpha}.\nabla q'(\underline{p}(0)) \leq p^*,$$

it follows from theorem 2.2 that

$$\nabla q'(\underline{p}(1)) - \nabla q'(\underline{p}(0)) \leq \alpha.(\underline{p}(1) - \underline{p}(0)),$$

thus

$$\nabla q'(\underline{p}(1)) \leq \alpha.(\underline{p}(1) - \underline{p}(0) + \frac{1}{\alpha}.\nabla q'(\underline{p}(0))) = \underline{p}.$$

Now suppose that for some $k \geq 1$

$$\underline{p}(0) \leq \underline{p}(k-1) \leq \underline{p}(k) \leq p^* \text{ and } \nabla q'(\underline{p}(k)) \leq \underline{p},$$

it follows from theorem 2.2 that

$$\nabla q'(p^*) - \nabla q'(\underline{p}(k)) \leq \alpha.(p^* - \underline{p}(k)),$$

since $\nabla q'(\underline{p}(k)) \leq \underline{p}$ and $\nabla q'(p^*) = \underline{p}$, we have

$$\underline{p}(0) \leq \underline{p}(k) \leq \underline{p}(k+1) = \underline{p}(k) - \frac{1}{\alpha}.\nabla q'(\underline{p}(k)) \leq p^*,$$

it follows from theorem 2.2 that

$$\nabla q'(\underline{p}(k+1)) - \nabla q'(\underline{p}(k)) \leq \alpha.(\underline{p}(k+1) - \underline{p}(k)),$$

thus

$$\nabla q'(\underline{p}(k+1)) \leq \alpha.(\underline{p}(k+1) - \underline{p}(k) + \frac{1}{\alpha}.\nabla q'(\underline{p}(k))) = \underline{p}.$$

Since $P = \{p \in R^n / \underline{p} \leq p \leq p^*\}$ is closed and bounded, the monotone increasing sequence $\{\underline{p}(k)\}$ is convergent. Thus, there exists $\hat{p} \in P$ such that $\lim_{k\to\infty} \underline{p}(k) = \hat{p}$ and by the continuity of the gradient mapping F, $\hat{p} = F(\hat{p})$. It follows from the convexity of the dual problem that a vector $p \in R^n$ is a fixed point of the gradient mapping F if and only if p is a solution of the dual problem. Hence, we have $\hat{p} = p^*$, since p^* is the minimal nonnegative element of P^*.

Q.E.D.

The gradient algorithm lends itself very well to distributed or parallel implementation. In particular each price p_i can be associated with a different processor.

We consider now totally asynchronous iterative algorithms (see [9, Section 6.1]). In brief, an asynchronous iterative algorithm relative to the solution of the fixed point problem $p = F(p)$ (where F is a mapping from R^n onto itself) is a sequence $\{p(k)\}$ of vectors of R^n defined as follows.

We assume that there is a set of times $T = \{0, 1, 2, ...\}$ at which one or more components p_i of p are updated by some processor. Let T^i be the set of times at which p_i is updated, we have for each $i = 1, ..., n$:

$$p_i(k+1) = F_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))), \forall k \in T^i, \tag{12}$$

$$p_i(k+1) = p_i(k), \forall k \notin T^i,$$

where F_i is the ith component of the mapping F, and for each $i = 1, ..., n$:

the set T^i is infinite,

$0 \leq \tau_j^i(k) \leq k$, $j = 1, ..., n$, $\forall k \in T^i$,

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 every j.

For further details about asynchronous iterative algorithms the reader is referred to [2], [4], and [9, Chapter 6].

THEOREM 3.2. Let assumptions of Section 2 hold. Asynchronous gradient algorithms defined by (12) (where F is the gradient mapping defined by (11)) and starting from an estimate $p(0) \in P = \{p \in R^n / \underline{p} \leq p \leq p^*\}$ converge to p^*.

Proof. In order to keep the proof of convergence concise, we show that the sufficient conditions of convergence of the Asynchronous Convergence Theorem of Bertsekas and Tsitsiklis (see [9, p. 431]) are satisfied.

Consider the sequence of nonempty sets $\{P(k)\}$ defined by

$$P(k) = \{p \in R^n / \underline{p}(k) \leq p \leq p^*\}, k = 0, 1, ...,$$

where the sequence $\{\underline{p}(k)\}$ is the gradient iteration defined by (10) and which starts from $\underline{p}$. We note that $P(k+1) \subset P(k) \subset P, k = 0, 1, ...$.

Clearly the sequence $\{P(k)\}$ satisfies the Box Condition: for every k, there exist sets $P_i(k) = \{p_i \in R / \underline{p}_i(k) \leq p_i \leq p_i^*\}$, $i = 1, ..., n$, such that $P(k)$ is the Cartesian product $P(k) = P_1(k) \times ... \times P_n(k)$.

By the isotonicity of the gradient mapping F on R^n and the proof of Theorem 3.1, the sequence $\{P(k)\}$ satisfies also the Synchronous Convergence Condition: $F(p) \in P(k+1)$, $\forall k$ and $p \in P(k)$ and $\{p(k)\}$ converges to p^* if $\{p(k)\}$ is a sequence such that $p(k) \in P(k)$ for every k.

Hence, the convergence result follows from the Asynchronous Convergence Theorem of Bertsekas and Tsitsiklis (see [9, p. 431]). Bertsekas and Tsitsiklis have shown that if the Synchronous Convergence and Box Conditions are satisfied and the starting point belongs to $P(0)$, then for each $k \geq 0$ there is a time after which all solution estimates given by the asynchronous algorithm are in $P(k)$ and all estimates used in (12) come from $P(k)$.

Q.E.D.

The convergence results of this paper can be extended without difficulties to scaled gradient algorithms and asynchronous scaled gradient algorithms relative to the scaled gradient mapping F' with components F'_i given by $F'_i(p) = p_i$, if $i = d$, and $F'_i(p) = p_i - \frac{1}{\alpha'_i} . \left.\frac{\partial q}{\partial p_i}\right|_p$, if $i \neq d$, where $\alpha'_i = \beta . a_i$, since F' is also isotone on R^n (see proof of theorem 2.2)

We note also that if for all $(i, j) \in A$, there exists a constant β_{ij} such that the subgradients of ∇c^*_{ij} at t_{ij} are less or equal to β_{ij}, $\forall t_{ij} \in R$, then the convergence results of this paper can also be extended to scaled gradient algorithms and asynchronous scaled gradient algorithms relative to the scaled gradient mapping F'' with components F''_i given by $F''_i(p) = p_i$, if $i = d$, and $F''_i(p) = p_i - \frac{1}{\alpha''_i} \cdot \frac{\partial q}{\partial p_i}\Big|_p$, if $i \neq d$, where $\alpha''_i = \sum_{(i,j) \in A} \beta_{ij} + \sum_{(m,i) \in A} \beta_{mi}$.

4. COMPUTATIONAL EXPERIENCE

We present now computational experiments on a distributed memory multiprocessor Tnode 16-32. The machine consists of a network of 16 to 32 processors T800, the so-called transputers, with some local memory. A transputer T800 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 machine 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...

Gradient and relaxation algorithms (denoted by G and R, respectively) were implemented on one processor, asynchronous and synchronous gradient and relaxation algorithms were implemented on 2, 4, 8, and 16 processors (they are denoted by AGx, SGx, ARx, and SRx, respectively, where x is the number of processors).

4.1. Problems considered

In this Section, we exhibit some classes of problems for which asynchronous gradient algorithms present good performances or which are on the contrary pathological cases. We concentrate first on problems for which nonlinear flow equations cannot be solved accurately and fast as a function of the end node prices. We will see in the sequel that asynchronous gradient algorithms present an interest in this case specially for problems with low node degrees. We conclude this subsection with a pathological case: quadratic transportation problems. In this preliminary computational experience we have considered only grid-like network flow problems. Although this network topology is somewhat simple and restrictive, it is nevertheless worthy of a performance study of parallel asynchronous gradient algorithms.

We have solved problems with the following three cost functions:

1) $c_{ij}(f_{ij}) = \frac{1}{2}\, f_{ij}^2$, if $-\,\frac{1}{2} \leq f_{ij} \leq \frac{1}{2}$, $c_{ij}(f_{ij}) = f_{ij}^4 + \frac{1}{16}$, if $f_{ij} < -\,\frac{1}{2}$, or $\frac{1}{2} < f_{ij}$;

In that case we have:

$\nabla c_{ij}^*(p_i - p_j) = p_i - p_j$, if $-\frac{1}{2} \leq p_i - p_j \leq \frac{1}{2}$, and $\nabla c_{ij}^*(p_i - p_j) = sign(p_i - p_j).(\frac{1}{4}.|p_i - p_j|)^{\frac{1}{3}}$ if $p_i - p_j \geq \frac{1}{2}$ or $p_i - p_j \leq -\frac{1}{2}$.

This cost function satisfies assumptions (a), (b), (c), and (d).

2) $c_{ij}(f_{ij}) = (\frac{1}{1-f_{ij}} + 0.5).f_{ij}$, if $0 \leq f_{ij} < 1$, and $c_{ij}(f_{ij}) = +\infty$, if $f_{ij} < 0$ or $1 \leq f_{ij}$;

In that case:

$\nabla c_{ij}^*(p_i - p_j) = 1 - (\frac{1}{p_i - p_j - \frac{1}{2}})^{\frac{1}{2}}$, if $p_i - p_j \geq 1.5$, and $\nabla c_{ij}^*(p_i - p_j) = 0$ if $p_i - p_j \leq 1.5$.

This type of cost function often found in communication problems satisfies also assumptions (a), (b), (c), and (d).

3) $c_{ij}(f_{ij}) = f_{ij}^2$, we have in this case $\nabla c_{ij}^*(p_i - p_j) = \frac{1}{2}.(p_i - p_j)$,

this quadratic cost function satisfies assumptions (a), (b), (c), and (d).

For all problems, there is only one nonzero traffic input, say b_1; $b_1 = 4$ for problems with the first cost function, $b_1 = 1$ for problems with the second and third cost functions. All problems have low node degrees ($\max_{i \in N} a_i = 4$).

For all problems and methods, the starting point is $p_i = 0$, for all $i \in N$. For the gradient methods we choose a stepsize:

$\alpha = \beta. \max_{i \in N} a_i = 4$ ($\beta = 1$ is chosen according to (d)), in the case of network flow problems with the first cost function;

$\alpha = 2$ ($\beta = 0.5$ chosen according to (d)), in the case of problems with the second and third cost functions.

The number of nodes and arcs varies from 18 to 144 and 27 to 237, respectively, for nonquadratic problems and from 24 to 1280 and 37 to 2476, respectively, for quadratic problems.

For nonquadratic problems line searches of the relaxation methods are stopped when $\frac{\partial q}{\partial p_i} \leq 0.001, \forall i \in N - \{d\}$. Computations of the gradient and relaxation methods are stopped when $\frac{\partial q}{\partial p_d} \leq \epsilon = 0.1$. We have shown in [20, proposition 4.2] that the sum of the absolute values of the partial derivatives of the dual functional over all nodes but the destination are less than ϵ if the termination test is satisfied. Thus this termination criterion can be used to detect global termination of sequential and parallel iterative algorithms. This remark is also true if we consider asynchronous implementation. Moreover this termination test presents the advantage to give a measure of the feasibility of the final solution since there is a direct connection between the partial derivatives of the dual functional and the conservation of flow constraints. The computational experiments have shown that the absolute values of the partial derivatives of the dual functional are in general very small compared with ϵ.

4.2. Implementation of parallel algorithms

For all algorithms and problems we try to balance the number of nodes associated with the different processors except for algorithms AG16, SG16, AR16, and SR16, in the case of problems with size 120, where 9 nodes are associated to even processors and 6 nodes are associated to odd processors. This case permits in particular to study the performances of parallel gradient algorithms for uneven load balancing.

We use a pipeline network of processors with bidirectional links. Figure 1 shows in particular the processes running on the different processors in the case of asynchronous implementation. This network topology seems naturally well suited to grid-like network flow problems when few processors are available. Task scheduling is made according to static mode.

4.2.1. Asynchronous implementation

The implementation was carried out in OCCAM, the transputer language based upon the communicating sequential processes notation. Occam provides only synchronous communication facilities. Thus for asynchronous implementation of iterative methods, we have proposed to implement two concurrent processes in parallel on each processor: a low level priority process, the so-called computation process, performs updatings and sends the updates to adjacent processors, a high level priority process, the so-called buffer process, stores data sent by adjacent processors and transmits the data to the computation process according to the requests sent by the computation process (see figure 1). The use of a buffer process permits to implement asynchronous communication and more generally to obtain asynchronous schemes of computation. The process computation iterates on the basis of the most recent data available in the buffer, in the beginning of each new updating. The process buffer which has very fast elementary processes has a higher level of priority than the process computation which consumes more time. Hence, the transmission of data from the computation process of a processor to the buffer process of its neighbor is not delayed. The buffer process is idle while it is waiting for messages. All the cpu time is then allocated to the process computation, because the scheduler of the transputer T800 is designed so that idle processes do not consume cpu time. A detailed description of asynchronous implementation of relaxation and gradient algorithms on a multitransputer system can be found in [18].

4.2.2. Synchronous implementation

In the case of synchronous iterative algorithms computation and data exchange are made sequentially. Processors communicate the updates after each computation step. Communication occurs only with adjacent processors in the pipeline. The data exchange process consists of the parallel input and output of updates. Processors are synchronized by message exchange, since communication is synchronized and not bufferised in Occam.

4.3. Computational results

Figures 2, 3, and 4 show the solution times in seconds for relaxation and gradient methods in function of the number of nodes in the network.

4.3.1. Nonquadratic problems

For nonquadratic problems price relaxation must be made by an iterative process which may consume nonnegligeable time. Figures 2 and 3 point out that G was generally faster than R. Moving in the good direction using a gradient step was more suitable than approximating with a good accuracy the prices that minimize the dual functional.

The speedups of the different parallel iterative methods are shown in tables 1 to 8. The nonlinear dual functional cannot be minimized analytically with respect to each price. Thus parallel relaxation algorithms lead to nondeterministic load unbalancing, since line search is made by an iterative procedure. Tables 3 and 7 show that an asynchronous implementation has speeded up efficiently the relaxation method. Tables 3, 4, 7, and 8 point out that synchronous relaxation methods were slower than asynchronous relaxation methods. In particular the synchronization penalties were great. We note that idle time due to synchronization may be great in this case since we have nondeterministic load unbalancing.

There is deterministic load balancing in the particular case of parallel gradient algorithms since we compute essentially a gradient at each updating and we have considered very regular network topologies: i.e. grid-like networks that can be partitioned and assigned equitably to the different processors. Tables 1 and 5 point out that an asynchronous implementation has speeded up very efficiently the gradient method. Asynchronous gradient algorithms were generally faster than synchronous gradient algorithms (in particular when more than two processors are used). The good performances of synchronous gradient algorithms shown on tables 2 and 6 are due in particular to the fact that we have considered essentially very regular network topologies i.e. grid-like networks, accordingly, idle time due to synchronization is very small. In the case of nonregular network topologies, idle time due to synchronization should be greater. From tables 1, 2, 5, and 6, we note that asynchronous implementation of the gradient method presents an interest particularly for unbalanced problems i.e. problem with 120 nodes and 16 processors. For the same ratio size of the problem over number of processors, the efficiency e, of asynchronous gradient algorithms has decreased slowly with the number of processors. For example we have obtained for problems with a cost function very similar to the first one: $e = 0.94$ for AG2 and problem of size 18, $e = 0.93$ for AG4 and problem of size 36, $e = 0.93$ for AG8 and problem of size 72. The efficiency of synchronous gradient algorithms has decreased faster with the number of processors: $e = 0.95$ for SG2 and problem of size 18, $e = 0.87$ for SG4 and problem of size 36, $e = 0.83$ for SG8 and problem of size 72.

We note also that AG2, AG4, and AG8 were faster than AR2, AR4, and AR8.

4.3.2. Quadratic transportation problems

Figure 4 shows that gradient algorithms are slower than relaxation algorithms for quadratic transportation problems. This is mainly due to the fact that price relaxation can be made analytically in this case, thus this process is not time consuming.

The value of the dual functional, solution time, number of iterations, and accuracy of the termination test are shown in table 9 for the relaxation algorithm applied to a quadratic transportation problem with 36 nodes and 57 arcs; we note that an additional digit is obtained in practice for the dual functional if an additional digit on the partial derivative at the destination node is required. The behaviour of gradient algorithms is very similar.

Table 10 shows the speedups of parallel relaxation an gradient algorithms for a quadratic transportation problem with 1280 nodes and 2476 arcs. We note that the speedups of parallel relaxation and gradient algorithms were very close. We recall that computational loads are very well balanced. However we point out that the speedups of asynchronous algorithms were generally better than the speedups of synchronous algorithms. In the case of nonregular network topologies, idle time due to synchronization should be greater. Thus the performances of parallel synchronous algorithms should be worse. Reference is made to [7], [10], and [32] for detailed computational experiments.

Obviously other pathological cases are reached for gradient algorithms when the degree of nodes becomes very large.

Algorithm / Size	AG2	AG4	AG8	AG16
48	1.92	3.75	7.17	-
72	1.97	3.88	7.52	-
96	1.98	3.92	7.65	14.61
120	1.99	3.95	7.75	14.60

Table 1 : speedups of AG2, AG4, AG8, and AG16 for problems with the first cost function

Algorithm / Size	SG2	SG4	SG8	SG16
48	1.92	3.65	6.67	-
72	1.97	3.80	7.14	-
96	1.98	3.85	7.33	13.41
120	1.98	3.90	7.49	12.11

Table 2 : speedups of SG2, SG4, SG8, and SG16 for problems with the first cost function

Algorithm / Size	AR2	AR4	AR8	AR16
48	1.70	3.13	6.70	-
72	1.72	3.18	6.24	-
96	1.73	3.24	6.24	12.84
120	1.74	3.27	6.30	12.93

Table 3 : speedups of AR2, AR4, AR8, and AR16 for problems with the first cost function

Algorithm / Size	SR2	SR4	SR8	SR16
48	1.51	2.14	2.71	-
72	1.55	2.38	3.33	-
96	1.56	2.54	3.81	5.09
120	1.57	2.65	4.17	6.42

Table 4 : speedups of SR2, SR4, SR8, and SR16 for problems with the first cost function

Algorithm / Size	AG2	AG4	AG8	AG16
48	1.84	3.47	6.50	-
72	1.87	3.54	6.73	-
96	1.88	3.55	6.81	12.78
120	1.88	3.54	6.78	12.50
144	1.87	3.53	6.79	12.90

Table 5 : speedups of AG2, AG4, AG8, and AG16 for problems with the second cost function

Algorithm / Size	SG2	SG4	SG8	SG16
48	1.83	3.35	6.49	-
72	1.84	3.39	6.37	-
96	1.84	3.37	6.22	11.78
120	1.83	3.32	6.09	9.87
144	1.83	3.22	5.89	11.02

Table 6 : speedups of SG2, SG4, SG8, and SG16 for problems with the second cost function

Size \ Algorithm	AR2	AR4	AR8	AR16
48	1.65	2.92	4.86	-
72	1.67	3.04	5.38	-
96	1.67	3.09	5.68	9.40
120	1.66	3.10	5.67	9.55
144	1.64	3.09	5.85	10.30

Table 7 : speedups of AR2, AR4, AR8, and AR16 for problems with the second cost function

Size \ Algorithm	SR2	SR4	SR8	SR16
48	1.47	2.53	3.89	-
72	1.46	2.57	4.44	-
96	1.45	2.54	4.61	7.50
120	1.44	2.51	4.57	7.26
144	1.44	2.49	4.57	8.21

Table 8 : speedups of SR2, SR4, SR8, and SR16 for problems with the second cost function

accuracy of the termination test	dual functional	iterations	time
0.1	11.04873	252	0.224
0.01	11.36527	486	0.430
0.001	11.39390	719	0.634
0.0001	11.39676	953	0.840
0.00001	11.39704	1187	1.046
0.000001	11.39707	1420	1.251

Table 9 : Accuracy of the relaxation algorithm for a quadratic problem

number of processors \ Algorithm	AG	SG	AR	SR
2	1.86	1.88	1.96	1.96
4	3.68	3.64	3.88	3.72
8	7.20	7.04	7.36	6.88
16	13.76	13.12	13.12	12.00

Table 10 : speedups of parallel algorithms for problem with 1280 nodes

5. CONCLUSIONS

In this paper we have proposed solving the dual of a strictly convex single commodity network flow problem by an original asynchronous gradient method. We have given a set of sufficient conditions for the convergence of gradient and asynchronous gradient algorithms. The convergence mechanism is original. Our result is different from the convergence result of Bertsekas for unconstrained optimization, which is based upon the diagonal dominance of the Hessian matrix of the cost function.

We have also presented and analysed computational results for parallel relaxation and gradient algorithms. The computational experience was carried out using a distributed memory multiprocessor. We have exhibited some good or pathological cases for asynchronous gradient algorithms. The experience has shown in particular that relaxation method is faster than gradient method for quadratic problems. On the contrary gradient methods can be interesting for nonquadratic problems with low-degree nodes. In that case nonlinear flow equations may not be solved particularly fast, moreover the gradient parameter is not small, thus using a gradient step may be more suitable than approximating with a good accuracy the prices that minimize the dual functional. The experience has shown that a distributed asynchronous implementation speeds up efficiently gradient and relaxation algorithms. The experience has also pointed out that the speedups are generally better for asynchronous algorithms than for synchronous algorithms.

Acknowledgments: The author wish to thank the reviewers for their helpful remarks.

References

[1] R. H. Barlow and D. J. Evans, "Synchronous and asynchronous iterative parallel algorithms for linear systems", *Comput. J.*, 25 (1982), 56-60.

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

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

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

[5] D. P. Bertsekas and D. A. Castanõn, "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).

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

[7] D. P. Bertsekas and D. A. Castañon, J. Eckstein, and S. Zenios, "Parallel computing in network optimization", report LIDS-P-2236, Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA, 1994, to appear in *Handbook on Operation Research and Management Science.*

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

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

[10] D. P. Bertsekas, P. Hossein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs", *SIAM Journal on Control and Optimization*, v. 25, 1987, 1219-1243.

[11] Y. Censor and A. Lent, "An iterative row-action method for interval convex programming", *Journal of Optimization Theory and Applications*, 34 (1981), 321-353.

[12] Y. Censor and J. Segman, "On block-iterative entropy maximization", *Journal of Information and Optimization Sciences*, 8 (1987), 275-291.

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

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

[15] R. De Leone et O. L. Mangasarian, "Asynchronous parallel successive overrelaxation for the symmetric linear complementarity problem", *Mathematical Programming*, B 42 (1988), 347-361.

[16] J. D. P. Donnely, "Periodic chaotic relaxation", *Linear Algebra Appl.* 4 (1971), 117-128.

[17] D. El Baz, "M-functions and parallel asynchronous algorithms", *SIAM Journal on Numerical Anal-*

ysis, 27 (1990), 136-140.

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

[19] D. El Baz, "Nonlinear systems of equations and parallel asynchronous iterative algorithms", in *Advances in Parallel Computing 9, Parallel Computing : Trends and Applications*, (North Holland, Amsterdam, 1994), G.R. Joubert et al. editors, 89-96.

[20] D. El Baz, P. Spiteri, and J. C. Miellou, "Distributed asynchronous iterative methods with order intervals for a class of nonlinear network flow problems", LAAS report 94244, June 1994.

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

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

[23] B. Lubachevsky and D. Mitra, "A chaotic asynchronous algorithm for computing the fixed point of nonnegative matrix of unit spectral radius", *J. Assoc. Comput. Mach.*, 33 (1986), 130-150.

[24] J. C. Miellou, "Algorithmes de relaxation chaotique à retards", *R.A.I.R.O.*, R_ 1 (1975), 55-82.

[25] 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), 233-236.

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

[27] 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.

[28] D. Mitra, "Asynchronous relaxations for the numerical solution of differential equations by parallel processors", *SIAM J. Sci. Stat. Comput.*, 8 (1987), 43-58.

[29] S. Nielsen and S. A. Zenios, "Massively parallel algorithms for singly constrained convex programs", *ORSA Journal on Computing*, 4 (1992).

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

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

[32] S.A. Zenios and R.A. Lasken, "Nonlinear network optimization on a massively parallel Connection Machine ", *Annals of Operation Research*, 14 (1988) 147-166.

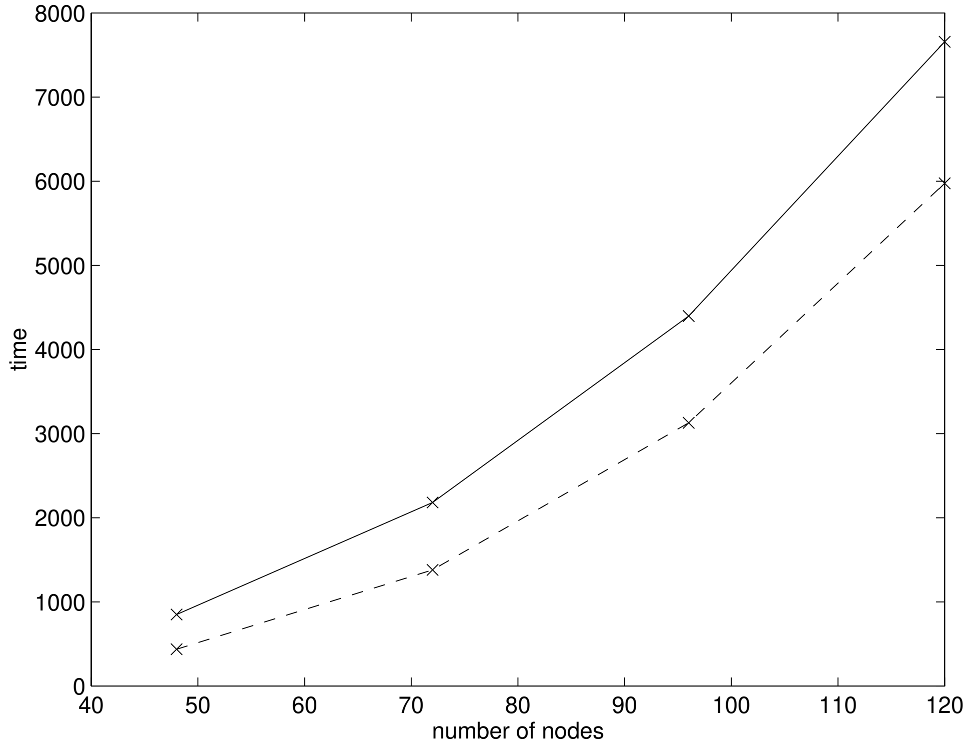

figure 2: times of R (solid) and G (dashed) for problems with the first cost function

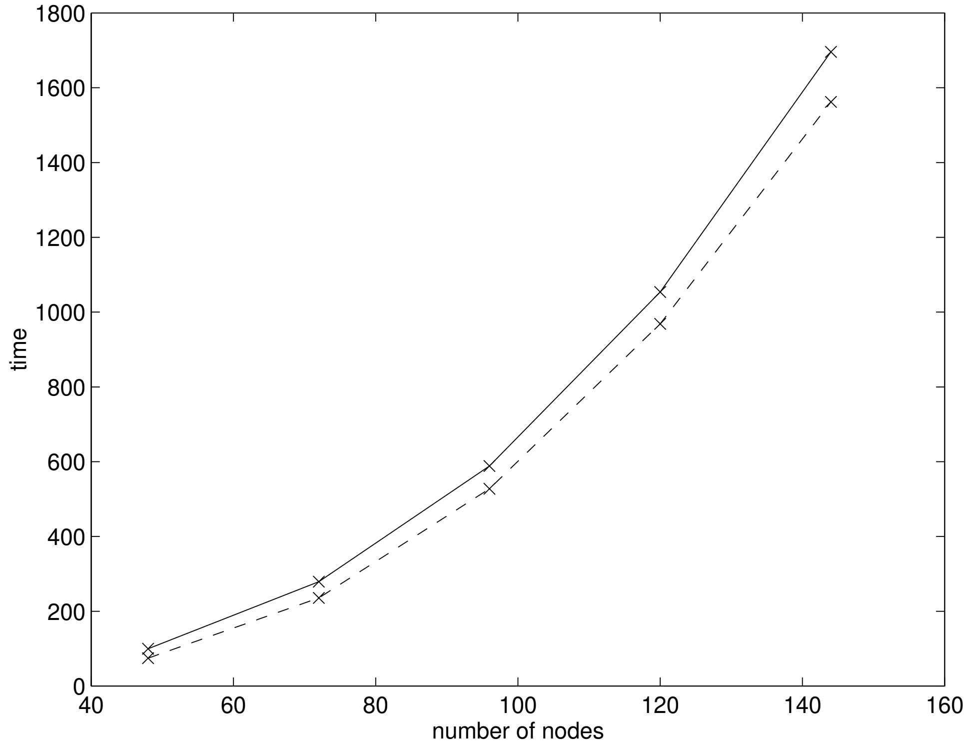

figure 3: times of R (solid) and G (dashed) for problems with the second cost function

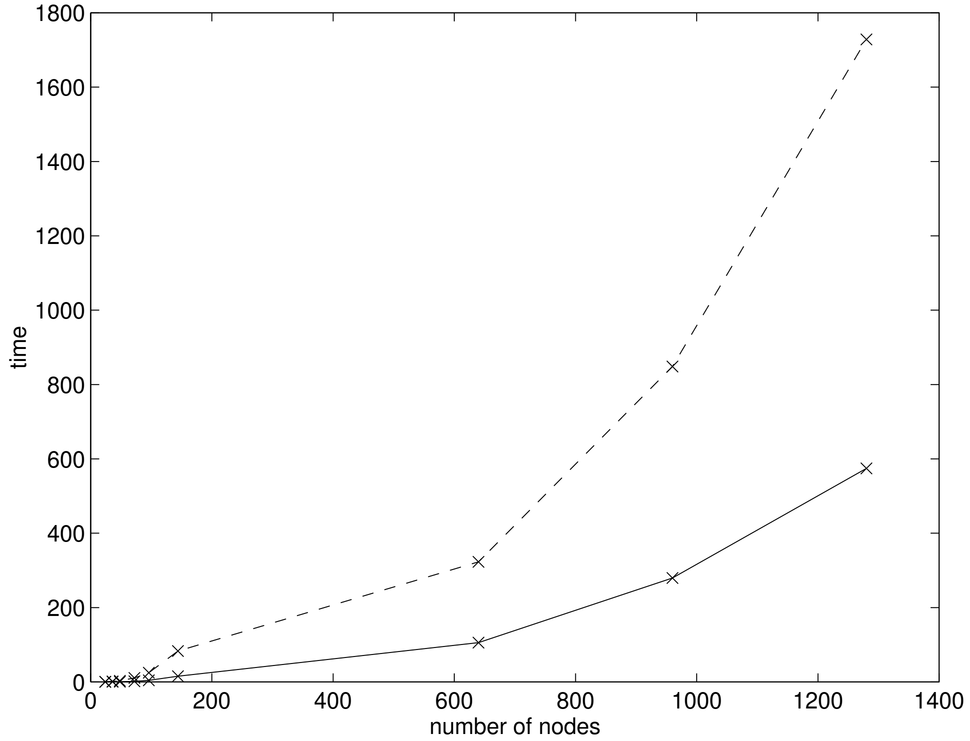

figure 4: times of R (solid) and G (dashed) for problems with the third cost function