2015 IEEE International Parallel and Distributed Processing Symposium Workshop

Parallel Asynchronous Modified Newton Methods for Network Flows

Didier El-Baz, Moussa Elkihel
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Université de Toulouse, LAAS, F-31400 Toulouse, France
Email: elbaz@laas.fr elkihel@laas.fr

***Abstract*—We consider single commodity strictly convex network flow problems. The dual problem is unconstrained, differentiable and well suited for solution via parallel iterative methods. We propose and prove convergence of parallel asynchronous modified Newton algorithms for solving the dual problem. Parallel asynchronous Newton multisplitting algorithms are also considered; their convergence is also shown. A first set of computational results is presented and analyzed.**

***Keywords*-Network flow problems, Newton method, multisplitting methods, parallel computing, asynchronous iterations.**

I. Introduction

Convex network flow problems occur in many fields like communication networks, water systems or gas distribution. Large scale convex network flow problems whose solution is time consuming occur frequently in real world applications. In this paper, we concentrate on the dual of the single commodity strictly convex network flow problem which is unconstrained, differentiable and well suited for solution via parallel iterative methods. This problem of great practical interest and has been studied for a long time, e.g., see [1] and [2].

In [3] and [4], respectively, we have shown that the structure of the dual problem allows the successful application of parallel asynchronous relaxation and gradient methods, respectively (see also [5]). In this paper, we present parallel asynchronous modified Newton methods for solving the dual problem.

Asynchronous iterative methods whereby iterations are carried out by several processors in arbitrary order and without any synchronization have been specially devised for parallel or 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. This contributes to make asynchronous iterations a general class of parallel iterative methods. The advantages of asynchronous iterative algorithms are computation flexibility, fault tolerance and tolerance to problem data changes. Since there is no synchronization overhead or idle time due to communication, one may also hope that parallel asynchronous implementation will be very efficient and scalable. There are many results in the literature relevant to the convergence of parallel or distributed asynchronous iterative methods. In particular, the reader is referred to [3], [4] and [6], for results in the optimization domain, and to [6], [7], [8], [9], [10], [11], [12] for results in the numerical simulation domain. In particular, we note that asynchronous gradient algorithms for unconstrained optimization are guaranteed to converge if the Hessian matrix of the cost function satisfies a diagonal dominance condition (see [6], Section 6.3).

Recently, the concern has risen on convergence rates of asynchronous iterations, see [13] (see also [14]). It is a challenging issue to design parallel asynchronous iterative methods for nonlinear optimization problems that have better convergence rates than parallel gradient algorithms. In this paper, we propose parallel asynchronous modified Newton methods and parallel asynchronous Newton multisplitting methods for convex network flow problems. We show that these methods converge. We present and analyze a first set of computational results for parallel asynchronous modified Newton methods applied to some communication problems and turbulent flow problems.

Section II deals with background material related to the convex network flow problem. In Section III, we propose and show convergence of parallel asynchronous modified Newton methods and parallel asynchronous Newton multisplitting methods. Issues related to initial approximation for parallel asynchronous modified Newton methods and Newton multisplitting methods are also presented in Section III. We present and analyze computational experiments carried out on a cluster in Section IV. Computational results are displayed and analyzed in Section IV. A simple termination criteria for asynchronous iterative methods is also displayed in this Section. Section V deals with conclusions.

II. Background material

We present first the mathematical formulation of convex separable network flow problems (see Fig. 1).

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

978-1-4673-7684-6/15 $31.00 © 2015 IEEE
DOI 10.1109/IPDPSW.2015.34

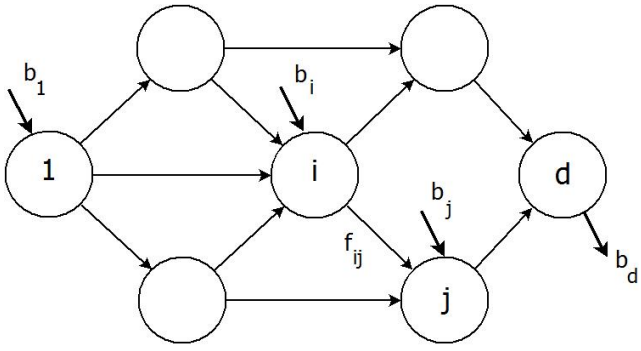

Fig. 1. Single destination network flow

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

subject to

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

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) $0 = arg\ min_{f_{ij}} c_{ij}(f_{ij})$;

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 [15], 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 [15], pp. 218, 253). We note that assumptions (a), (b) and (c) are not overly restrictive and are naturally satisfied in many practical situations. Below are few examples of cost functions that satisfy assumptions (a), (b) and (c).

- $c_{ij}(f_{ij}) = a_{ij}.|f_{ij}| + b_{ij}.f^2_{ij}$, with $a_{ij} \geq 0$ and $b_{ij} > 0$;
- $c_{ij}(f_{ij}) = a_{ij}.max\{f^2_{ij}, f^4_{ij}\}$, 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 $b_{ij} \geq 0$.

A dual problem for (1) is given by

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

subject to no constraint on the vector $p = \{p_i/i \in \mathcal{N}\}$, where q is the dual functional given by

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

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 (1) and (3) is explored in great detail in [1]. The necessary and sufficient condition for optimality of a pair (f, p) is given in [15]. A feasible flow vector $f = \{f_{ij}/(i,j) \in \mathcal{A}\}$ is optimal for (1) and a price vector $p = \{p_i/i \in \mathcal{N}\}$ is optimal for (3) if and only if for all arcs $(i,j) \in \mathcal{A}$,

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

An equivalent condition is

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

Any one of these equivalent relations is referred to as the complementary slackness condition (see [15] pp. 337-338 and [3]).

Existence of an optimal solution of the dual problem can be guaranteed under the following additional regular feasibility assumption (see [1] p. 360 and p. 329): there exists a feasible flow vector, $f = \{f_{ij}/(i,j) \in \mathcal{A}\}$, such that $c'_{ij-}(f_{ij}) < +\infty$ and $c'_{ij+}(f_{ij}) > -\infty$, for all $(i,j) \in \mathcal{A}$, 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

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

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 $i \in \mathcal{N} - \{d\}$, 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 $(i,j) \in \mathcal{A}$, $p_i - p_j$ is a subgradient of c_{ij} at f^*_{ij}, where $f^* = \{f^*_{ij}/(i,j) \in A\}$ is the unique primal optimal solution.

Theorem 1: Let assumptions of Section 2 hold. The intersection, denoted by I, of P^* with the nonnegative orthant is nonempty.

Proof: (see [4]). ■

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 $(i,j) \in \mathcal{A}$,

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 $i \in \mathcal{N}$, $p_i^* \leq p_i$.

In the sequel, we will assume that p_d is a constant, $p_d = 0$ and p will denote a vector in R^{n-1}. Throughout the paper the component-wise partial ordering on R^{n-1} will be written as $p^* \leq p$.

We consider the following nonlinear system of equations.

$$\nabla q(p) = 0, \tag{8}$$

where $\nabla q(p) = \{ \frac{\partial q}{\partial p_i}\Big|_p, i \in \mathcal{N} - \{d\}\}$. In the following, the number of arcs incident to node i is denoted by δ_i.

III. Asynchronous modified Newton methods

A. Modified Newton methods

In this subsection, we propose a first modified Newton method. Given a vector $p(0) \in R^{n-1}$, that is an initial approximation of the solution, the modified Newton method generates a sequence of vectors as follows.

$$p(k+1) = p(k) + d(k), k = 0, 1, ..., \tag{9}$$

where $d(k)$ is solution of the system

$$(H(p(k)) + \Delta(p(k)))d(k) = -\nabla q(p(k)), \tag{10}$$

$H(p)$ is the Hessian of function $q(p)$ which is defined as follows.
For all $i \in \mathcal{N} - \{d\}$,

$$\begin{aligned} \frac{\delta^2 q}{\delta p_i^2}\Big|_p &= \sum_{(i,j)\in\mathcal{A}} \nabla^2 c_{ij}^*(p_i - p_j) \\ &+ \sum_{(m,i)\in\mathcal{A}} \nabla^2 c_{mi}^*(p_m - p_i), \end{aligned} \tag{11}$$

for all $i, j \in N - \{d\}$,

$$\begin{aligned} \frac{\delta^2 q}{\delta p_i \delta p_j}\Big|_p &= -\nabla^2 c_{ij}^*(p_i - p_j) \text{ if } (i,j) \in \mathcal{A}, \\ &= -\nabla^2 c_{ji}^*(p_j - p_i) \text{ if } (j,i) \in \mathcal{A}, \\ &= 0 \text{ otherwise.} \end{aligned} \tag{12}$$

and $\Delta(p(k))$ is an $n-1 \times n-1$ diagonal matrix such that $H(p(k)) + \Delta(p(k))$ is positive definite (we may have $\Delta(p(k)) = 0$, whenever $H(p(k))^{-1}$ exists and $H(p(k))$ is positive definite). We note also that most of the time it is sufficient to add a small positive value to the diagonal entries of matrix $H(p(k))$ in order to ensure that $(H(p(k))+\Delta(p(k)))^{-1}$ exists and $H(p(k)) + \Delta(p(k))$ is positive definite.

Finally, we note that it follows from the definition of $\Delta(p(k))$ that $(H(p(k)) + \Delta(p(k)))^{-1}\nabla q(p(k))$ is always a descent direction.

In the sequel, we shall use the following notation

$$\mathcal{H}(p) = H(p) + \Delta(p). \tag{13}$$

We define the modified Newton mapping $\mathcal{G} : R^{n-1} \rightarrow R^{n-1}$ such that

$$\mathcal{G}(p) = p - (\mathcal{H}(p))^{-1}\nabla q(p). \tag{14}$$

We note that the solution of the system

$$\mathcal{H}(p(k))d(k) = \nabla q(p(k)), \tag{15}$$

may be particularly prohibitive when n is large. In this case, one can use an iterative method in order to obtain an approximate solution of the system (15). Let D be the diagonal of matrix $\mathcal{H}(p(k))$. We have

$$\mathcal{H}(p(k)) = D - D + \mathcal{H}(p(k)). \tag{16}$$

One can use for example a Jacobi iterative method based on the splitting (16). The vector $d(k)$ is then approximated by using the following Jacobi iterative scheme

$$Dd(k)^{t+1} = (D - \mathcal{H}(p(k)))d(k)^t + \nabla q(p(k)), t = 1, ..., v, \tag{17}$$

where v is a given positive integer and $d(k)^0 = 0$.

B. Parallel asynchronous algorithms

In this subsection, we present the solution of the dual problem via several parallel asynchronous modified Newton methods. Reference is made to [3], [4], [6], [16], [17], and [18], for various sequential and parallel iterative methods applied to equality and interval-constrained problems.

We propose parallel Newton algorithms which use multisplitting techniques in order to approximate the solution of system (15). For more references on mutisplitting techniques see [19], [20], [21], [22], [23], [24] and the references therein.

Recall that p is a $n-1$ dimensional unknown vector. In the sequel, we will concentrate on a multisplitting whereby vector p is partitioned into m subvectors. Consider m disjoint subsets I_l of $\{1, ..., n-1\}$ where $\{I_l\}_{1\leq l\leq m}$ is a partition of $\{1, ..., n-1\}$. Here, m denotes the number of parallel tasks. A task l consists in computing $card(I_l)$ components among the $n-1$ components of vector p.

Define for $p \in R^{n-1}$, the $n-1 \times n-1$ matrices $M_l(p)$ as follows.

$(M_l(p))_{ij} = (\mathcal{H}(p))_{ij}$ *if* $i, j \in I_l$,
$(M_l(p))_{ij} = (\mathcal{H}(p))_{ij}$ *if* $i \in I_r, j \in I_s, 1 \leq r < l$ and $1 \leq s < l$ or $l < r \leq m$ and $l < s \leq m$,
$(M_l(p))_{ij} = 0$ elsewhere.

Remark 1: The matrices $M_l(p)$ are block-diagonal matrices with nonnull entries equal to the entries of matrix $\mathcal{H}(p)$. They can be represented according to the following two patterns

$$(M_l(p)) = \overbrace{\left(\begin{array}{|c|c|}\hline \mathcal{H}^{11}(p) & 0 \\ \hline 0 & \mathcal{H}^{22}(p) \\ \hline \end{array}\right)}^{pattern1} \text{ or } \overbrace{\left(\begin{array}{|c|c|c|}\hline \mathcal{H}^{11}(p) & 0 & 0 \\ \hline 0 & \mathcal{H}^{22}(p) & 0 \\ \hline 0 & 0 & \mathcal{H}^{33}(p) \\ \hline \end{array}\right)}^{pattern2} \tag{18}$$

where pattern 1 is relative to the case where $l = 1$ or $l = m$ and pattern 2 corresponds to the case where $1 < l < m$.

Consider first pattern 1. Note that in the case where $l = 1$, $\mathcal{H}^{11}(p)$ is the $card(I_1) \times card(I_1)$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{11}_{ij}(p) = \mathcal{H}_{ij}(p)$ and $\mathcal{H}^{22}(p)$ is the $n-1-card(I_1) \times n-1-card(I_1)$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{22}_{ij}(p) = \mathcal{H}(p)_{card(I_1)+i,card(I_1)+j}$. In the case where $l = m$, $\mathcal{H}^{11}(p)$ is the $n-1-card(I_m) \times n-1-card(I_m)$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{11}_{ij}(p) = \mathcal{H}_{ij}(p)$ and $\mathcal{H}^{22}$ is the $card(I_m) \times card(I_m)$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{22}_{ij}(p) = \mathcal{H}(p)_{n-1-card(I_m)+i,n-1-card(I_m)+j}$.

Consider now Pattern 2, i.e. $1 < l < m$. In this case, $\mathcal{H}^{22}(p)$ is the $card(I_l) \times card(I_l)$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{22}_{ij}(p) = \mathcal{H}(p)_{\sum_{t=1}^{l-1} card(I_t)+i,\sum_{t=1}^{l-1} card(I_t)+j}$. Similarly, $\mathcal{H}^{11}(p)$ is the $\sum_{t=1}^{l-1} card(I_t(p)) \times \sum_{t=1}^{l-1} card(I_t(p))$ submatrix of $\mathcal{H}(p)$ such that $\mathcal{H}^{11}_{ij}(p) = \mathcal{H}_{ij}(p)$. The matrix $\mathcal{H}^{33}(p)$ is defined accordingly.
Define also matrices $N_l(p)$ such that $N_l(p) = M_l(p) - \mathcal{H}(p), 1 \leq l \leq m$. Then we have a family of splittings

$$\mathcal{H}(p) = M_l(p) - N_l(p), 1 \leq l \leq m. \tag{19}$$

Define now diagonal nonnegative weighting matrices $E_l(p)$, $1 \leq l \leq m$, such that

$$(E_l(p))_{ii} = 1 \text{ for all } i \in I_l, \ (E_l(p))_{ii} = 0 \text{ elsewhere.} \tag{20}$$

Since $\{I_l\}_{1 \leq l \leq m}$ is a partition of $\{1, ..., n-1\}$, we have

$$\sum_{l=1}^{m} E_l(p) = I, \tag{21}$$

where I denotes the identity matrix. In this way, for each p, the familly of splittings $\{M_l(p), N_l(p), E_l(p)\}_{l=1}^m$ is a multisplitting of $\mathcal{H}(p)$.

The solution of system (15) is approximated by performing v iterations of the multisplitting method starting from an initial approximation $d(k)^0 = 0$. Thus, we have

$$d(k)^1 = \sum_{l=1}^{m} E_l(p(k)) M_l(p(k))^{-1} \nabla q(p(k)), \tag{22}$$

$$\begin{aligned} d(k)^2 &= \sum_{l=1}^{m} E_l(p(k)) M_l(p(k))^{-1} N_l(p(k)) d(k)^1 \\ &+ M_l(p(k))^{-1} \nabla q(p(k))), \end{aligned} \tag{23}$$

and so on. With this particular multisplitting, the l-th parallel task only needs to compute the value of the components of the l-th subvector of p.

The parallel modified Newton multisplitting method can be defined recursively as follows.

$$p(k+1) = G(p(k)), \tag{24}$$

where

$$G(p) = p - A(p) \nabla q(p), \tag{25}$$

and

$$A(p) = \sum_{l=1}^{m} E_l(p) \sum_{j=0}^{v-1} (M_l(p)^{-1} N_l(p))^j M_l(p)^{-1}, \tag{26}$$

is obtained by a simple calculus relative to the multisplitting scheme starting with $d^0 = 0$. We have also

$$A(p) = A(p)\mathcal{H}(p))\mathcal{H}(p))^{-1}. \tag{27}$$

It follows from (27), (19) and (26) that

$$\begin{aligned} A(p) &= \sum_{l=1}^{m} E_l(p) \sum_{j=0}^{v-1} (M_l(p)^{-1} N_l(p))^j \mathcal{H}(p))^{-1} \\ &- \sum_{l=1}^{m} E_l(p) \sum_{j=1}^{v} (M_l(p)^{-1} N_l(p))^j \mathcal{H}(p))^{-1}. \end{aligned} \tag{28}$$

Thus, we have

$$\begin{aligned} A(p) &= \sum_{l=1}^{m} E_l(p) (M_l(p)^{-1} N_l(p))^0 \mathcal{H}(p))^{-1} \\ &- \sum_{l=1}^{m} E_l(p) (M_l(p)^{-1} N_l(p))^v \mathcal{H}(p))^{-1}. \end{aligned} \tag{29}$$

and

$$A(p) = \sum_{l=1}^{m} E_l(p) (I - (M_l(p)^{-1} N_l(p))^v (\mathcal{H}(p))^{-1}. \tag{30}$$

We assume now that there exist a unique solution $p^* \in P^*$. For example, there exists a unique solution if the cost functions c_{ij} are continuously differentiable for all $(i,j) \in \mathcal{A}$ (see[25]).

Theorem 2: Let assumptions of Section 2 hold and assume that there exists a unique optimal solution p^*. Then, there exists a ball $B(p^*, r)$ of center p^* and radius r, such that the parallel Newton multisplitting methods defined by (24), (25), (30) and starting from an estimate $p(0) \in B(p^*, r)$, converge to p^*.

Proof: It follows from (11), (12), (13) and the definition of $\Delta(p)$ that the matrices $\mathcal{H}(p)$ are M-matrices. Thus, for all $p \in R^{n-1}$ there exists a positive vector $u \in R^{n-1}_+$ such that $\mathcal{H}(p)u > 0$. By definition of the multisplitting $\{M_l(p), N_l(p), E_l(p)\}_{l=1}^m$, we have clearly $M_l(p)u > 0$, $l = 1, ..., m$. Thus, the matrices $M_l(p)$, $l = 1, ..., m$ are M-matrices and as a consequence we have $M_l^{-1}(p) > 0$, $l = 1, ..., m$. Moreover, by construction of the multisplitting $\{M_l(p), N_l(p), E_l(p)\}_{l=1}^m$, the matrices $N_l(p)$ are positive. It follows that the splittings $M_l(p) - N_l(p)$, $l = 1, ..., m$ are regular splittings (see [20] and [23]) and the result follows from Theorem 3 in [19]. ■

We consider now parallel asynchronous iterative algorithms (see [6] Section 6.1). In brief, a parallel asynchronous iterative algorithm relative to the solution of the fixed point problem $p = G(p)$ (where G is a mapping from R^{n-1} onto itself) is a sequence $\{p(k)\}$ of vectors of R^{n-1} defined as follows.
Let the iterate vector p be decomposed into m subvectors p_l, $l = 1, ..., m$, where p_l will denote in the remaining of this Section the subvector relative to the subset I_l, i.e. the subvector with components associated with elements of I_l.

Assume that there is a set of times $T = \{0, 1, 2, ...\}$ at which the components of one or several subvectors of the iterate vector are updated by some processor. Let T^l be the subset of times at which the components of the l-th subvector are updated. We have for each $l \in \{1, ..., m\}$,

$$p_l(k+1) = G_l(p_1(\tau_1^l(k)), ..., p_m(\tau_m^l(k))), \forall k \in T^l, \quad (31)$$

$$p_l(k+1) = p_l(k), \forall k \notin T^l,$$

where G_l is the l-th block-component of the mapping G and for each $l \in \{1, ..., m\}$,

- (d) the set T^l is infinite,
- (e) $0 \leq \tau_j^l(k) \leq k$, $j \in \{1, ..., m\}$, $\forall k \in T^l$,
- (f) if $\{k_t\}$ is a sequence of elements of T^l that tends to infinity, then $\lim_{t\to\infty} \tau_j^l(k_t) = +\infty$, for every $l \in \{1, ..., m\}$.

The above conditions ensure respectively that no component of the iterate vector is abandoned forever during the updating process and old values of the components of the iterate vector are replaced by new updates as the computation goes on. For further details about asynchronous iterative algorithms the reader is referred to [3], [6], [7], [8], [9], [10] and [26].

Theorem 3: Let assumptions of Section 2 hold and assume that P^* has a unique optimal solution p^*. Then, there exists a ball $B(p^*, r)$ of center p^* and radius r, such that for all initial estimate $p(0) \in B(p^*, r)$, asynchronous Newton multisplitting algorithms defined by (25), (30), (31) and satifying assumptions (d), (e) and (f) converge to p^*.

Proof: It follows from (25) and $\nabla q(p^*) = 0$ that we have

$$G'(p^*) = I - A(p^*)\mathcal{H}(p^*). \quad (32)$$

Moreover, it follows from (30) that we have

$$G'(p^*) = I - \sum_{l=1}^{m} E_l(p^*)(I - (M_l(p^*)^{-1}N_l(p^*))^v. \quad (33)$$

It results from the definition of the weighting matrices that

$$G'(p^*) = \sum_{l=1}^{m} E_l(p^*)(M_l(p^*)^{-1}N_l(p^*))^v. \quad (34)$$

As shown in the proof of Theorem 2, the splittings $M_l(p^*) - N_l(p^*)$, $l = 1, ..., m$ are regular splittings (see [20]). Consider now the Jacobi matrix of $\mathcal{H}(p^*)$ which is denoted by $M'^{-1}(p^*)N'(p^*)$ where the matrix $M'(p^*)$ is a diagonal matrix with diagonal entries equal to the diagonal entries of matrix $\mathcal{H}(p^*)$. We have

$$\rho\{M'^{-1}(p^*)N'(p^*)\} < 1, \quad (35)$$

since $\mathcal{H}(p^*)$ is an M-matrix. The splitting $M'(p^*) - N'(p^*)$ is also clearly a regular splitting. Moreover, if A is an M-matrix and if $A = B_1 - C_1 = B_2 - C_2$ are two regular splittings of A, then (see [27]))

$$C_2 \leq C_1 \Rightarrow \rho\left(B_2^{-1}C_2\right) \leq \rho\left(B_1^{-1}C_1\right),$$

the strict inequality holds if $C_2 \neq C_1$and $B_1^{-1}C_1$ is irreducible. Thus, if we compare the family of splittings $M_l(p^*) - N_l(p^*)$, $l = 1, ..., m$ to $M'(p^*) - N'(p^*)$ i.e. the splitting relative to the Jacobi method, we obtain by construction of the multisplitting $\{M_l(p^*), N_l(p^*), E_l(p^*)\}_{l=1}^m$, and by using the above result

$$\rho\{M_l^{-1}(p^*)N_l(p^*)\} < 1, l = 1, ..., m. \quad (36)$$

It follows from (34), (36) and Proposition 3.2 in [20] that we have

$$\rho\{G'(p^*)\} \leq \max_{l=1,...,m} \rho\{(M_l^{-1}(p^*)N_l(p^*))^v\} < 1. \quad (37)$$

Thus, there exists a neighborhood of p^*, denoted by $V(p^*)$, where the modified Newton multisplitting mapping G is P-contracting in p^*. We consider the ball of center p^* and radius r, denoted by $B(p^*, r)$ such that $B(p^*, r) \subset V(p^*)$ and the result follows from (37) and Theorem 3.11 in [28] (see also [22], Theorem 4.4). ■

Remark 2: The choice of $M_l(p)$ to be the diagonal elements of $\mathcal{H}(p)$, i.e. typically the Jacobi method, would be the worse choice in term of number of iterations. We see also that if $\overline{M}_l(p) \geq M_l(p)$, then the splitting $\{\overline{M}_l(p), \overline{N}_l(p), \overline{E}_l(p)\}$ is better than the splitting $\{M_l(p), N_l(p), E_l(p)\}$, the condition $\overline{M}_l(p) \geq M_l(p)$ implies that the iterations are more implicit, thus the solution of the corresponding linear system requires more time. So, in order to have better performance, one has to choose block-multisplittings which make a good compromise between the number of iterations and the duration of a typical iteration.

We conclude this subsection with some results concerning parallel asynchronous modified Newton methods whereby the fixed point mapping $\mathcal{G}$ is defined by (14). We assume that there exists a unique solution $p^* \in P^*$. In this case, we have

$$\mathcal{G}'(p^*) = \frac{-\nabla q(p^*)\mathcal{H}'(p^*)}{\mathcal{H}(p^*)^2} = 0. \quad (38)$$

Thus, we have

$$\rho\{M(\mathcal{G}'(p^*))\} = 0 < 1. \quad (39)$$

Then, there exists a neighborhood of p^*, denoted by $V(p^*)$ where the modified Newton mapping $\mathcal{G}$ is P-contracting in p^*. We consider also the ball of center p^* and radius r, denoted by $B(p^*, r)$ such that $B(p^*, r) \subset N_{p^*}$. We have the following local convergence result for asynchronous modified Newton algorithms.

Theorem 4: Let assumptions of Section 2 hold and assume that P^* has a unique optimal solution p^*. Then, there exists a ball $B(p^*, r)$ of center p^* and radius r, such that for all initial estimate $p(0) \in B(p^*, r)$, asynchronous modified Newton algorithms defined by (31), where the fixed point mapping $\mathcal{G}$ is the modified Newton mapping defined by (14), and satifying assumptions (d), (e) and (f) converge to the unique optimal solution p^*.

Proof: The proof follows from (39) and Theorem 3.11 in [28]. ■

C. Initial approximation

In subsection III-B, we have shown the local convergence of parallel asynchronous modified Newton methods and parallel asynchronous Newton multisplitting methods. In this subsection, we recall some results that permit one to generate good initial approximations for parallel modified Newton methods and parallel Newton multisplitting methods. In particular, we note that the parallel Newton methods quoted above can be combined with parallel gradient algorithms. In that case, parallel asynchronous (or synchronous) gradient algorithms start with an initial approximation of the solution that can be far from the solution and deliver an approximation of the solution in the domain of convergence of asynchronous (or synchronous) modified Newton methods and asynchronous (or synchronous) Newton multisplitting methods. Nevertheless, the following assumption must be added (see [4]).

- (g) there exists a constant $\beta \geq 0$, such that for all $(i,j) \in \mathcal{A}$ 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 [1] p. 320).

The gradient iteration is defined by

$$p(k+1) = p(k) - \frac{1}{\alpha}.\nabla q(p(k)), k = 0,1,..., \qquad (40)$$

where $\alpha = \beta.\max_{i\in N}\delta_i.$ and δ_i is the degree of node i. The gradient mapping $F : R^{n-1} \to R^{n-1}$ is given by

$$F(p) = p - \frac{1}{\alpha}.\nabla q(p). \qquad (41)$$

Theorem 5: Under the hypotheses of Section 2 and assumption (g), there exists a constant $\alpha = \beta.\max_{i\in N}\delta_i$ such that for all $p, p'' \in R^{n-1}$, with $p'' \leq p$, we have

$$\nabla q(p) - \nabla q(p'') \leq \alpha.(p - p''). \qquad (42)$$

Proof: See [4]. ∎

It follows clearly from the above theorem that F is isotone on R^{n-1} (i.e. whatever $p, p' \in R^{n-1}$, $p' \leq p$ implies $F(p') \leq F(p)$). Since for all $(i,j) \in \mathcal{A}$, c^*_{ij} is real valued and continuously differentiable, it follows from (6) that for all $i \in \mathcal{N} - \{d\}$, $\frac{\partial q}{\partial p_i}$ is continuous on R^{n-1}. Hence, ∇q and F are also continuous on R^{n-1}.

The gradient algorithm lends itself very well to parallel or distributed synchronous and asynchronous implementation. We consider now asynchronous gradient algorithms (see [4]) according to model (31) where we substitute G for F, Then we have the following result.

Theorem 6: Let assumptions of Section 2 hold. Asynchronous gradient algorithms defined by (31), (41) that satisfies assumptions (d), (e) and (f) and start from an estimate $p(0) \in P = \{p \in R^n / \underline{p} \leq p \leq p^*\}$ converge to p^*.

Proof: See [4]. ∎

IV. Computational experience

Computational tests have been carried out on a cluster with 16 processors. Parallel synchronous and asynchronous modified Newton methods relative to fixed point mapping (14) that start with an initial approximation generated via parallel synchronous and asynchronous gradient methods, respectively, were implemented on up to 16 processors.

A. Parallel implementation

4.1.1. Synchronous case

In the synchronous case, updating and data exchanges are performed sequentially. Processors communicate the updates at the end of each updating phase. Processors are synchronized by message exchanges. Communications are implemented via the MPI Library. More precisely, each processor sequentially sends prices value by using the MPI-Isend() function and receive data by using MPI-Recv() function. We note that the function MPI-Isend() is nonblocking and that synchronization is realized via the MPI-Recv() function which is blocking.

4.1.2. Asynchronous case

Asynchronous implementation was mainly achieved by using the put() function of the SHMEM library. The put() function permits one processor (the source) to make a copy from a part of its own memory in the memory of a target processor. We note that the put() function allows to cover communications by computations since processors are not blocked till the completion of the communication.

B. Problems

The problems considered in the computational tests are network flow problems with 96 nodes and 144 nodes. The network flow problems have been decomposed so as to balance the computational load as fairly as possible amongst the different processors. This was performed by equilibrating the number of nodes in the network on the different processors. For a given size of problems, we have also studied cases with various topologies. We have concentrated on cases where the maximum degree of a node can be equal to 4, 12, and 22, respectively. These cases correspond to contexts that generate different task granularities and permit one to study the impact of task granularity on the efficiency of parallel methods.

We have considered communication problems with cost functions given as follows:

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

In that case we have

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

This type of cost function satisfies assumptions (a), (b), (c) in Section II and assumption (g) in Section III (see [4]).

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

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

We have taken $b = 1.85$; this case corresponds to turbulent flow in pipes (see [29], [30] and [31]). In that case, we have

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

This cost function satisfies also assumptions (a), (b), (c) and assumption (g) on a bounded subdomain (see [18]).

For communication problems, the traffic input is given by $b_i \geq 0.01$ for all $i \in \mathcal{N} - \{d\}$ and $b_d = -\sum_{i \in \mathcal{N}-\{d\}} b_i$. For turbulent flow problems, we have considered the case where there are only two nonzero traffic input, say $b_1 = 1$ and $b_d = -b_1 = -1$.

For all problems and methods, the initial approximation is $p_i = 0$, for all $i \in \mathcal{N}$.

The stepsize of the gradient methods is given by $\alpha = \beta . \max_{i \in \mathcal{N}} \delta_i$, where δ_i denotes the degree of node i and β is chosen in order to satisfy assumption (d); $\beta = 0.5$ in the case of communication problems and $\beta = 0.73$ in the case of turbulent flows.

Computations of the gradient methods are stopped when $\frac{\partial q}{\partial p_d} \leq \epsilon = 10^{-4}$. We have shown in Proposition 4.2 of [18] 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 this termination test is satisfied. Thus, this termination criterion can be used to detect global termination of sequential and parallel iterative algorithms (this remark is valid in the asynchronous context). Moreover, this termination test presents the advantage to give a measure of the feasibility of the solution since the partial derivatives of the dual functional are directly related to 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 as compared with ϵ. Computations of the modified Newton methods are stopped when $\frac{\partial q}{\partial p_d} \leq \epsilon' = 10^{-13}$.

C. Computational results

The efficiencies of parallel synchronous and asynchronous modified Newton methods are displayed in Figs 2 to 5 where async.-22 is the efficiency of parallel asynchronous modified Newton methods for problems with maximum node degree equal to 22 and sync.-22 is the efficiency of parallel synchronous modified Newton methods for problems with maximum node degree equal to 22. Figs 2 to 5 show that the greater the size of the problem and the maximum degree of nodes, i.e., the greater the task granularity, the better is the efficiency. Figs 2 to 5 show also that asynchronous modified Newton methods are more efficient than synchronous modified Newton methods. Finally, we note that parallel modified Newton methods, lead to deterministic load balancing since we have considered very regular network topologies that give rise to fair partitioning, whereby nodes are equitably assigned to the different processors.

Computational results clearly show the interest of parallel asynchronous modified Newton methods. As an example, turbulent flow problems with size 144 and maximum node degree equal to 22 have been solved via parallel asynchronous modified Newton methods in less than 30 seconds on a cluster with 16 processors (the sum of the absolute values of the deficits at each node but the destination, i.e., the errors on the conservation of flow constraints, is less than $\epsilon' = 10^{-13}$). Experimental results also show the impact of task granularity on the efficiency of the parallel methods. As a consequence, large scale network flow problems can be solved via parallel asynchronous modified Newton methods with satisfactory efficiency.

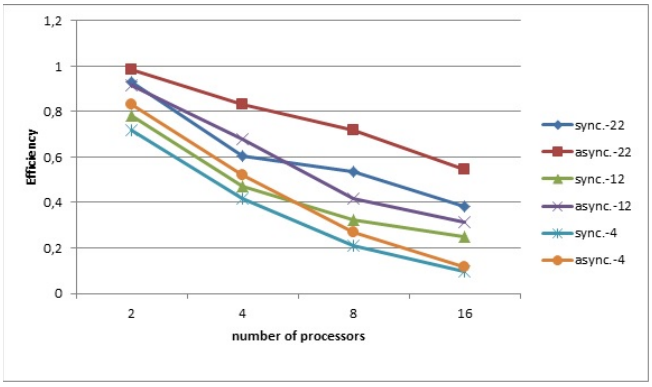

Fig. 2. Efficiency of synchronous and asynchronous Newton multisplitting methods in function of the number of processors for communication problems with size 96 and maximum node degree equal to 4, 12 and 22.

V. Conclusions

In this paper, we have proposed to solve the dual of a strictly convex network flow problem via original parallel asynchronous modified Newton methods and Newton multisplitting methods. We have shown the local convergence of these methods. We have also presented a stopping criteria specially designed for convex network flow problems.

We have presented and analyzed computational results for parallel synchronous and asynchronous iterative schemes of computation. Computational results show the interest of parallel asynchronous modified Newton methods. In general, parallel asynchronous modified Newton methods are more efficient than synchronous modified Newton methods. Computational results have also shown the impact of task granularity on the efficiency of the considered iterative schemes.

Clearly, parallel asynchronous modified Newton methods converge faster than gradient methods or relaxation methods that were previously considered in the literature for the solution of convex network flow problems. These methods are well suited to large scale problems; they are also scalable provided a minimum task granularity is considered.

References

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

[2] D. Bertsekas, *Network Optimization continuous and discrete models*. Athena Scientific, Belmont Massachusetts, 1998.

[3] D. Bertsekas and D. El Baz, "Distributed asynchronous relaxation methods for convex network flow problems," *SIAM J. on Control and Optimization, vol. 25*, pp. 74–85, 1987.

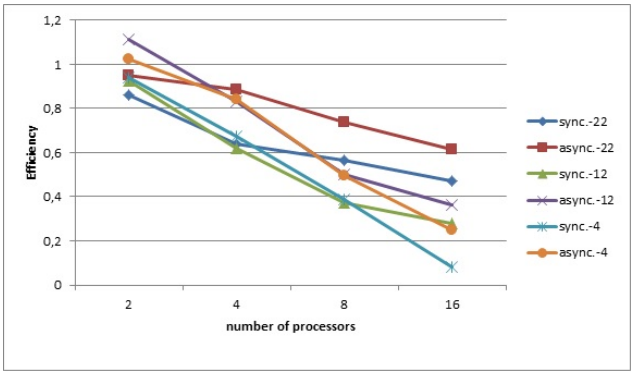

Fig. 3. Efficiency of synchronous and asynchronous Newton multisplitting methods in function of the number of processors for communication problems with size 144 and maximum node degree equal to 4, 12 and 22.

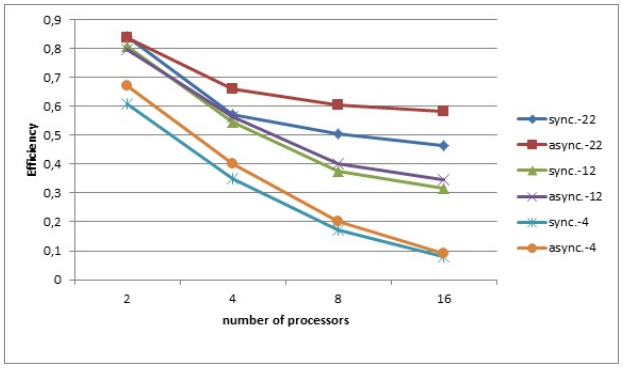

Fig. 4. Efficiency of synchronous and asynchronous Newton multisplitting methods in function of the number of processors for turbulent flow problems with size 96 and maximum node degree equal to 4, 12 and 22.

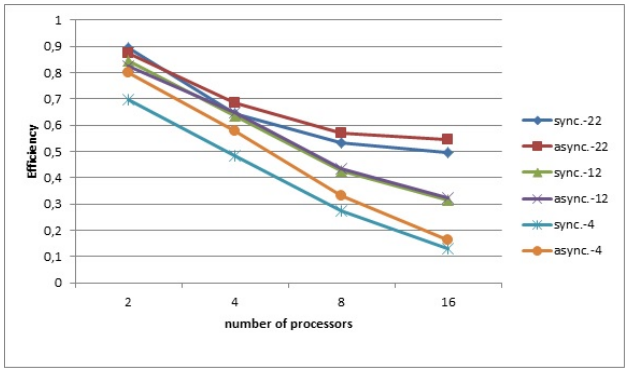

Fig. 5. Efficiency of synchronous and asynchronous Newton multisplitting methods in function of the number of processors for turbulent flow problems with size 144 and maximum node degree equal to 4, 12 and 22.

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

[5] ——, "A computational experience with distributed asynchronous iterative methods for convex network flow problems," in *Proceedings of the 28th IEEE Conference on Decision and Control*, 1989, pp. 590—591.

[6] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. (republished in 1997 by Athena Scientific), 1989.

[7] A. Chazan and W. Miranker, "Chaotic relaxation," *Linear Algebra Applications*, pp. 437–449, 1969.

[8] J.-C. Miellou, "Algorithmes de relaxation chaotique à retards," *RAIRO, R1*, pp. 55–82, 1975.

[9] G. M. Baudet, "Asynchronous iterative methods for multiprocessors," *J. Assoc. Comput. Mach., 2*, pp. 226–244, 1978.

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

[11] J.-C. Miellou, D. El Baz, and P. Spiteri, "A new class of asynchronous iterative algorithms with order intervals," *Mathematics of Computation, Vol. 67, n. 221*, pp. 237–255, 1998.

[12] P. Spiteri, J.-C. Miellou, and D. El Baz, "Perturbation of parallel asynchronous linear iterations by floating point errors," *Electronic Transactions on Numerical Analysis, Vol. 13*, pp. 38–55, 2002.

[13] R. Feyzmahdavian and M. Johansson, "On the convergence rates of asynchronous iterations," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014.

[14] M. Tutunov, R.and Zargham and A. Jadbabaie, "On convergence rate of accelerated dual descent algorithm," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014, pp. 179–184.

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

[16] D. P. Bertsekas, P. Hossein, and P. Tseng, "Relaxation methods for network flow problems with convex arc cost," *SIAM Journal on Control and Optimization, Vol. 25, Issue 5*, pp. 1219–1243, 1987.

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

[18] D. El Baz, P. Spiteri, J.-C. Miellou, and D. Gazen, "Asynchronous iterative algorithms with flexible communication for nonlinear network flow problems," *Journal of Parallel and Distributed Computing, 38*, pp. 136–140, 1996.

[19] J. Arnal, V. Migallon, and J. Penadés, "Nonstationary parallel Newton iterative methods for nonlinear problems," in *Lectures Notes in Comput. Science, VECPAR'2000*, 2001, pp. 380—394.

[20] J. Bahi, J. C. Miellou, and K. Rhofir, "Asynchronous multisplitting methods for nonlinear fixed point problems," *Numerical Algorithms, Vol. 15, n. 3*, pp. 315–345, 1997.

[21] J. Bahi, K. Rhofir, and J. C. Miellou, "Parallel solution of linear DAEs by mutisplitting waveform relaxation methods," *Linear Algebra and applications, Vol. 332-334*, pp. 181–196, 2001.

[22] A. Frommer and D. Szyld, "On asynchronous iterations," *Journal of Computational and Applied Mathematics*, pp. 237–255, 2000.

[23] D. O'Leary and R. White, "Multi-splittings of matrices and parallel solution of linear systems," *SIAM Journal on algebraic discrete methods, 6*, pp. 630–640, 1985.

[24] P. Spiteri, J.-C. Miellou, and D. El Baz, "Parallel asynchronous Schwarz and multisplitting methods for nonlinear diffusion problems," *Numerical Algorithms*, pp. 437–449, 1995.

[25] D. El Baz, "Asynchronous iterative algorithms for convex network flow problems," in *Proceedings of the first European Control Conference*, Grenoble, France, 1991, pp. 2397—2402.

[26] D. Bertsekas, "Distributed asynchronous computation of fixed points," *Mathematical Programming, 27*, pp. 107–120, 1983.

[27] J. M. Ortega and W. C. Rheinbold, *Iterative solutions of nonlinear equations in several variables*. Academic Press, New York, 1970.

[28] M. N. El Tarazi, "Some convergence results for asynchronous algorithms, 39," *Num. Math.*, pp. 325–340, 1982.

[29] G. Birkhoff and J. B. Diaz, "Nonlinear network problems," *Quart. Appl. Math. 13*, pp. 431–443, 1965.

[30] T. Porshing, "Jacobi and Gauss-Seidel methods for nonlinear network problems," *SIAM J. Numer. Anal., 6*, pp. 437–449, 1969.

[31] W. Rheinboldt, "On M-functions and their application to nonlinear Gauss-Seidel iterations and to network flows," *Mathematical Analysis and Applications, 32*, pp. 274–307, 1970.