Computational Optimization and Applications, 6, 1–32 (1997)

Decomposition methods for differentiable optimization problems over Cartesian product sets

MICHAEL PATRIKSSON mipat@mai.liu.se
Department of Mathematics, University of Washington, Seattle, WA; presently at Department of Mathematics, Linköping University, Linköping, Sweden

Editor: William W. Hager

Abstract. This paper presents a unified analysis of decomposition algorithms for continuously differentiable optimization problems defined on Cartesian products of convex feasible sets. The decomposition algorithms are analyzed using the framework of cost approximation algorithms. A convergence analysis is made for three decomposition algorithms: a sequential algorithm which extends the classical Gauss–Seidel scheme, a synchronized parallel algorithm which extends the Jacobi method, and a partially asynchronous parallel algorithm. The analysis validates inexact computations in both the subproblem and line search phases, and includes convergence rate results. The range of feasible step lengths within each algorithm is shown to have a direct correspondence to the increasing degree of parallelism and asynchronism, and the resulting usage of more outdated information in the algorithms.

Keywords: Cartesian product sets, decomposition, cost approximation, sequential algorithm, parallel processing, partially asynchronous algorithms

1. Introduction

The impact of parallel and distributed computers on the practice of scientific computations has been significant. Mathematical programming within operations research was among the first scientific areas where the potential of parallel numerical computing was recognized, and is a field on which the advent of parallel computing has had a deep influence. As the number, size and complexity of computational problems continue to grow, we can expect a continuing development of faster computers, new computer architectures, and designated chips for the most important applications; in order to take full advantage of this development, the area of parallel algorithm design will however need to mature with at least the same rate.

One may distinguish between two approaches to the design of a parallel algorithm: it is either constructed by parallelizing an existing (or, suitably modified) serial algorithm, or it is designed from scratch, perhaps with a particular computer architecture in mind. In this paper we take the first approach, and consider a class of differentiable optimization problems whose structure can be exploited in the design of parallel algorithms. More specifically, we consider the application of cost approximation (CA) algorithms to the minimization of continuously differentiable functions over convex product sets, and study their basic convergence characteristics when adapted to various computing models. This class of iterative algorithms includes many classical methods for differentiable optimization, and its convergence behaviour has been well studied in its serial version; the analysis made here supplies information about what can be expected when parallelizing an existing serial CA code for the problem under consideration.

1.1. The problem under study

The point in common to the problems we consider is that the feasible set is a *Cartesian product* of sets, that is, we assume that the feasible set X can be described by

$$X = \prod_{i \in \mathcal{C}} X_i, \qquad X_i \subseteq \Re^{n_i}, \qquad \sum_{i \in \mathcal{C}} n_i = n, \tag{1}$$

for some finite index set $\mathcal{C}$, where each set X_i is nonempty, closed and convex. The variables associated with these sets are therefore independent, but they are (implicitly) grouped together through the objective function $f : X \mapsto \Re$, which is assumed to be continuously differentiable on an open set containing X ($f \in C^1$ on X for short). We consider the constrained differentiable problem

[CDP]

$$\underset{x \in X}{\text{minimize}}\ f(x).$$

One example of a constraint structure of the form (1) is that of (generalized) box constraints,

$$X_i = \{\, x_i \in \Re \mid a_i \leq x_i \leq b_i \,\}, \qquad -\infty \leq a_i \leq b_i \leq +\infty, \qquad i \in \{1, 2, \ldots, n\};$$

unconstrained optimization is included in this framework. The structure (1) is also inherent in many applications of equilibrium programming, in which case $\mathcal{C}$ denotes an index set of players in a non-cooperative game (e.g., a Nash equilibrium game) [31], commodities of goods in a market or spatial price equilibrium problem [48], or pairs of origins and destinations in a transportation network [56]. The last application of these is discussed briefly below.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ denote a network of nodes and directed links. A subset $\mathcal{C}$ of $\mathcal{N} \times \mathcal{N}$ defines a set of commodities, associated with pairs of origins and destinations of trips. Denoting by h_{pqr}, $r \in \mathcal{R}_{pq}$, the flow on route r between the nodes p and q, the total flow on a link $a \in \mathcal{A}$ is

$$x_a = \sum_{(p,q) \in \mathcal{C}} \sum_{r \in \mathcal{R}_{pq}} \delta_{pqra} h_{pqr},$$

where

$$\delta_{pqra} = \begin{cases} 1, & \text{route } r \text{ passes through link } a, \\ 0, & \text{otherwise.} \end{cases}$$

Associated with each link $a \in \mathcal{A}$ is a positive and strictly increasing function $c_a : \Re_+ \mapsto \Re_+$ of the total flow on the link. The *traffic assignment problem* (or, the *traffic equilibrium problem*) is to find a set of consistent route and link flows solving

[TAP]

$$\text{minimize } f(x) = \sum_{a \in \mathcal{A}} \int_0^{x_a} c_a(s)\, ds,$$

subject to

$$\begin{aligned} \sum_{r \in \mathcal{R}_{pq}} h_{pqr} &= d_{pq}, \qquad \forall (p,q) \in \mathcal{C}, \\ \sum_{(p,q) \in \mathcal{C}} \sum_{r \in \mathcal{R}_{pq}} \delta_{pqra} h_{pqr} &= x_a, \qquad \forall a \in \mathcal{A}, \\ h_{pqr} &\geq 0, \qquad \forall (p,q) \in \mathcal{C}. \end{aligned}$$

That [TAP] is of the form [CDP] is revealed if the link flow definitional constraints are eliminated from the problem, in which case it turns into a problem in the route flow variables h_{pqr} only. To give an idea of the size of a practical problem, we mention that in one application [22] the number of nodes and links is about $10,000$ and $30,000$, respectively, and the total number of origin and destination nodes 1000 (in which case potentially the number of commodities is about one million). More common, however, is a problem size in the order of 1000, 3000 and $10,000$ in the number of nodes, links and commodities, respectively.

A problem with a structure identical to that of [TAP] arises in message routing problems in computer communication networks [26]. For a derivation and overview of [TAP], see [56].

Some additional applications are mentioned next.

A recent trend in the modelling of traffic problems is to use supplementary (or, side) constraints; these are introduced in order to improve the quality of an existing model by incorporating additional information, such as link flow observations or link flow capacity constraints originating from a centralized traffic control, or to derive the link tolls that should be introduced to reach some traffic management goal without imposing a centralized traffic control [38, 39].

Decentralized decision-making problems, where the subsystems share a scarce resource, give rise to optimization models with *block-angular* constraint structures [2, 20, 64, 60, 65]. This type of planning problems arise, for example, in transportation and distribution management and project planning and scheduling [40, 2, 21].

Most successful methods of attack for problems of the form described above involve a (augmented) Lagrangean dualization of the coupling (or, side) constraints; see, for example, [40, 2, 21, 11, 64, 66, 69]. The subproblems in any such *relaxation* scheme will be problems of the form [CDP]. The idea of utilizing Lagrangean dualization as a means to decomposition of a mathematical program was formulated at least as early as 1960 ([19]; see also [40, 21]). Several transformations of optimization problems that enhance parallelism can be found in [11, Sec. 3.4]; most of them are based on Lagrangean duality, and they all result in a problem of the form [CDP].

The above examples illustrate that applications of [CDP] are more wide-spread than what is perhaps apparent at first sight.

A few words on the notation used: all vectors are column vectors, the inner product of x and y in $\Re^n$ is denoted by $x^{\mathrm{T}}y$, and its associated vector norm by $\|\cdot\|$. Corresponding to the partition of $\Re^n$ defined by the product (1), we let x_i denote the subvector of variable components corresponding to index $i \in \mathcal{C}$. Further, we let i_- and i_+ denote the sets of indices less than i and greater than i, respectively. [We adopt the convention that i_- (i_+) is empty for $i = 1$ ($i = |\mathcal{C}|$).] Thus, we introduce the partitioning $x^{\mathrm{T}} = (x_1^{\mathrm{T}}, \dots, x_{|\mathcal{C}|}^{\mathrm{T}})$, where $x_i \in \Re^{n_i}$ and $\sum_{i \in \mathcal{C}} n_i = n$; for convenience, we shall frequently write $x^{\mathrm{T}} = (x_{i_-}^{\mathrm{T}}, x_i^{\mathrm{T}}, x_{i_+}^{\mathrm{T}})$ for some $i \in \mathcal{C}$. For a function $f : X \mapsto \Re^n$, we let $\nabla_i f$ denote the partial derivative of f with respect to the component x_i. A vector-valued function $F : X \mapsto \Re^n$ is said to be *monotone* on X if

$$[F(x) - F(y)]^{\mathrm{T}}(x - y) \geq 0, \qquad \forall x, y \in X,$$

strictly monotone on X if this holds with strict inequality whenever $x \neq y$, and *strongly monotone* on X (with modulus $m_F > 0$) if

$$[F(x) - F(y)]^{\mathrm{T}}(x - y) \geq m_F \|x - y\|^2, \qquad \forall x, y \in X;$$

it is *Lipschitz continuous* on X (with modulus $M_F > 0$) if

$$\|F(x) - F(y)\| \leq M_F \|x - y\|, \qquad \forall x, y \in X.$$

The *indicator* function and *normal cone* operator for X is denoted by δ_X and N_X, respectively; the *subdifferential*, *conjugacy*, and *inverse* operator is denoted, respectively, by ∂, $^\circ$, and $^{-1}$. We denote *effective domain* and *interior* by `dom` and `int`, respectively. A point–to–set map $A : X \mapsto 2^X$ is said to be *closed* at $x \in X$ if $\{x^t\} \to x$, $y^t \in A(x^t)$ and $\{y^t\} \to y$ imply that $y \in A(x)$; it is *upper semicontinuous* (u.s.c.) at $x \in X$ if given a neighbourhood $N(A(x))$ of

$A(x)$ there exists a neighbourhood $N(x)$ of x such that $A(y) \subset N(A(x))$ whenever $y \in N(x) \cap X$. The map A is closed (respectively, u.s.c.) on a subset S of X if the above holds for all $x \in S$. A mapping A is said to be *maximal monotone* if it is monotone and its graph is not properly contained in the graph of any other monotone operator. The *lower level set* of f at $x^0 \in X$ with respect to X is $L(x^0) = \{ x \in X \mid f(x) \le f(x^0) \}$.

1.2. The cost approximation algorithm

In this paper, we shall develop and analyze iterative algorithms for the solution of [CDP], which are adapted to the Cartesian product structure of its constraints. The basis for the development is the convergence analysis of the *cost approximation (CA) algorithm* [54, 55] and the closely related family of *partial linearization algorithms* [52, 53]. These are designed to find stationary points of [CDP], that is, points $x^* \in X$ satisfying the *variational inequality*

[VIP]

$$\nabla f(x^*)^{\mathrm{T}}(x - x^*) \ge 0, \qquad \forall x \in X.$$

We let Ω denote the set of stationary points of [CDP]. (A stationary point is a global minimizer of f on X whenever f is pseudoconvex [4, Sec. 3.5].)

One iteration of a CA algorithm is defined by the following two steps:

(i) Given a point in $X \setminus \Omega$, a feasible search direction is defined through the solution of an approximation of [VIP], in which ∇f is approximated by a monotone mapping.

(ii) The direction defined by the solution to the above described subproblem is a feasible direction of descent with respect to f. A step is taken in this direction in order to reduce the value of f.

The first step is called the *cost approximation step*; the approximation is derived as follows.

Let $x \in X$. We introduce a continuous and monotone *cost approximating mapping* $\Phi : X \mapsto \Re^n$. If the function ∇f is replaced by Φ, then the error made obviously is $\nabla f - \Phi$. This error is taken into account by adding to Φ the fixed error term $\nabla f(x) - \Phi(x)$. Thus, the resulting subproblem is a monotone variational inequality problem of finding $y(x) \in X$ satisfying

[VIP$_\Phi$]

$$[\Phi(y(x)) + \nabla f(x) - \Phi(x)]^{\mathrm{T}}(y - y(x)) \ge 0, \qquad \forall y \in X.$$

We let $Y(x)$ denote the solution set of [VIP$_\Phi$]. (Note that the addition of a fixed vector to Φ does not affect the form of [VIP$_\Phi$].)

If Φ is chosen as the gradient mapping of a function $\varphi : X \mapsto \Re$, which then is convex and in C^1 on X, then [VIP$_\Phi$] is the optimality conditions for the convex subproblem

[CDP$_\varphi$]

$$\underset{y \in X}{\text{minimize}}\ f_\varphi(y) = \varphi(y) + f(x) - \varphi(x) + [\nabla f(x) - \nabla\varphi(x)]^{\mathrm{T}}(y - x).$$

The construction of this subproblem may be given an interpretation in terms of partial linearizations of f: if, in [CDP], f is replaced by φ, then the error is $f - \varphi$; linearizing this error at $x \in X$ yields the subproblem [CDP$_\varphi$].

The CA algorithm is adapted to the structure of [CDP] by letting the mappings Φ be separable with respect to the partition of $\Re^n$ given by (1), that is, by letting

$$\Phi(x)^{\mathrm{T}} = [\Phi_1(x_1)^{\mathrm{T}}, \ldots, \Phi_{|\mathcal{C}|}(x_{|\mathcal{C}|})^{\mathrm{T}}], \qquad x \in X, \tag{2}$$

where $\Phi_i : X_i \mapsto \Re^{n_i}$ is continuous and monotone on X_i. Then [VIP$_\Phi$] separates into $|\mathcal{C}|$ independent problems of finding a $y_i(x) \in X_i$ such that

[VIP$_{\Phi_i}$]

$$[\Phi_i(y_i(x)) + \nabla_i f(x) - \Phi_i(x_i)]^{\mathrm{T}}(y_i - y_i(x)) \geq 0, \qquad \forall y_i \in X_i.$$

If $\Phi_i \equiv \nabla\varphi_i$ for some convex function $\varphi_i : X_i \mapsto \Re$ in C^1 on X_i, then [VIP$_{\Phi_i}$] is the optimality conditions for

[CDP$_{\varphi_i}$]

$$\underset{y_i \in X_i}{\operatorname{minimize}}\; f_{\varphi_i}(y_i) = \varphi_i(y_i) + [\nabla_i f(x) - \nabla\varphi_i(x_i)]^{\mathrm{T}} y_i.$$

In the convergence analysis in [52, 53, 54, 55] of the CA algorithm, two forms of the cost approximating mapping have been used. The most general one is a sequence $\{\Phi^t\}$ of continuous and monotone mappings on X; each mapping may be chosen based on the current iterate x^t (or, more generally, on the whole sequence of iterates up to x^t), and there is neither an assumption of a dependency among the different mappings nor an assumption of the existence of a limit mapping. Less generally, the mappings are assumed to vary continuously with the iterates, that is, there exists a known continuous function $\Phi : X \times X \mapsto \Re^n$ which is monotone on X in its first argument; each individual cost approximating mapping Φ^t in the sequence $\{\Phi^t\}$ then has the form $\Phi(\cdot, x^t)$.

We next provide some important properties of the subproblems [VIP$_\Phi$] and [CDP$_\varphi$], and of the resulting search directions, that we will frequently be referring to in the subsequent analysis.

LEMMA 1 [54, 55] (Properties of the subproblem and search direction) *Let $x \in X$, $Y(x)$ be the (possibly empty) set of solutions to* [VIP$_\Phi$]*, $y \in Y(x)$, and $d = y - x$.*

(a) (A fixed point characterization of Ω)

  (i) $x \in \Omega \Longleftrightarrow x \in Y(x)$.

  (ii) *Let $\Phi \equiv \nabla\varphi$. Then, $x \in \Omega \Longleftrightarrow x \in Y(x) \Longleftrightarrow f_\varphi(y) = f_\varphi(x)$.*

(b) (A characterization of $Y(x)$) *Let Φ be maximal monotone. Then,*

$$Y(x) = [\Phi + N_X]^{-1}[\Phi - \nabla f](x).$$

(c) (A well-posedness characterization)

  (i) *Let Φ be maximal monotone. Then, the set $Y(x)$ is nonempty and bounded if*

$$\Phi(x) - \nabla f(x) \in \mathtt{int}\,(\mathtt{dom}\,([\Phi + N_X]^{-1})). \tag{3}$$

  (ii) *Let $\Phi \equiv \nabla\varphi$. Then, the set $Y(x)$ is nonempty and bounded if and only if*

$$\nabla\varphi(x) - \nabla f(x) \in \mathtt{int}\,(\mathtt{dom}\,([\varphi + \delta_X]^\circ)). \tag{4}$$

(d) (Closedness of the search direction mapping) *Let $\Phi : X \times X \mapsto \Re^n$ be a continuous mapping on $X \times X$ of the form $\Phi(y, x)$, monotone on X with respect to y. Then, the search-direction-defining mapping $x \mapsto D(x) = Y(x) - x$ is closed on X.*

(e) (Descent properties of d) *Let* $x \notin \Omega$.

(i) *If* Φ *is strictly monotone on* X, *then* $\nabla f(x)^{\mathrm{T}} d < 0$.

(ii) *Let* $\Phi \equiv \nabla\varphi$. *Let* $\overline{y} \in X$ *be any point such that* $f_\varphi(\overline{y}) < f_\varphi(x)$. *Then,* $\nabla f(x)^{\mathrm{T}}(\overline{y}-x) < 0$. *Especially,* $\nabla f(x)^{\mathrm{T}} d < 0$.

(iii) *Let* Φ *be strongly monotone on* X. *Let* $\overline{y} \in X$ *be an approximate solution to* [VIP$_\Phi$] *in the sense that for some scalar* $\varepsilon \geq 0$,

$$[\Phi(\overline{y}) + \nabla f(x) - \Phi(x)]^{\mathrm{T}}(y - \overline{y}) \geq -\varepsilon, \qquad \forall y \in X,$$

and $\overline{d} = \overline{y} - x$. *If* $\varepsilon < m_\Phi \|\overline{d}\|^2$, *then* $\nabla f(x)^{\mathrm{T}} \overline{d} < 0$. *Especially,* $Y(x)$ *is nonempty and singleton, and* $\nabla f(x)^{\mathrm{T}} d \leq -m_\Phi \|d\|^2$.

Remark. The fixed point property in (a) provides a termination criterion for the algorithm.

A sufficient condition for the set $Y(x)$ to be both nonempty and bounded for every $x \in X$ is that Φ is *coercive* on X [77], that is, that either X is bounded or

$$\lim_{\substack{\|x\| \to +\infty \\ x \in X}} \frac{\Phi(x)^{\mathrm{T}}(x - z)}{\|x\|} = +\infty,$$

for some $z \in X$; the corresponding condition for φ is that X is bounded or

$$\lim_{\substack{\|x\| \to \infty \\ x \in X}} \frac{\varphi(x)}{\|x\|} = +\infty.$$

(The coercivity property implies that the effective domains appearing in the right–hand–sides of (3) and (4), respectively, equal $\Re^n$; a strongly monotone mapping is coercive.)

A function $\varphi : X \times X \mapsto \Re$ of the form $\varphi(y, x)$, continuous on $X \times X$ and convex and differentiable on X with respect to y, automatically has the continuity property of the mapping $\Phi = \nabla_y \varphi$ required in the result (d).

The mapping Φ is maximal monotone if it is defined on $\Re^n$.

The result (e).ii and (e).iii validates inexact solutions of the subproblem [CDP$_\varphi$] and [VIP$_\Phi$], respectively.

The convergence analysis will reveal that the more general the forms of the cost approximating mappings are the stronger monotonicity properties necessarily have to be imposed on them. (The choice of cost approximating mappings as gradients somewhat lessens the monotonicity requirements.) On the other hand, with stronger monotonicity properties, the flexibility with which step lengths in the second main phase of the iterative algorithm may be chosen increases, and simultaneously the computational burden decreases. (This is important since a line search is a serial operation in general, which imposes a synchronization penalty in a parallel algorithm.)

In the second main stage of the CA algorithm, we consider three rules for choosing the step length ℓ_t in the update $x^{t+1} = x^t + \ell_t d^t$; the first one (the exact line search) is conceptual only, but the other two (the Armijo step length rule and the relaxation step) are implementable, and are validated below.

Definition 1 (Step length rules) *Let* $x \in X$, $y \in Y(x)$, *and* $d = y - x$.

(E) (Exact line search) *Choose* ℓ *as a solution to*

$$\text{minimize}\{\, f(x + \ell d) \mid \ell \geq 0; \quad x + \ell d \in X \,\}.$$

(A) (Armijo [1] step length) *Let* $\alpha,\ \beta \in (0,1)$. *Choose* $\ell = \beta^{\bar{\imath}}$, *where* $\bar{\imath}$ *is the smallest nonnegative integer* i *such that*

$$f(x+\beta^i d) - f(x) \le \alpha\beta^i \nabla f(x)^{\mathrm{T}} d. \tag{5}$$

(R) (Relaxation step length) *Assume that* ∇f *is Lipschitz continuous on* X, *and let* Φ *be strongly monotone on* X. *Further, let* $\varepsilon \in (0, m_\Phi/M_{\nabla f}]$. *Choose* ℓ *in* $[\varepsilon, 2m_\Phi/M_{\nabla f} - \varepsilon]$ *such that* $x + \ell d \in X$.

LEMMA 2 [54, 55] (Validity of the step length rules) *Let* $x \in X \setminus \Omega$, $y \in Y(x)$, *and* $d = y - x$.

(A) (i) *Let* Φ *be strictly monotone on* X. *Then, the inequality (5) holds for a finite integer* i.

(ii) *Let* $\Phi \equiv \nabla\varphi$. *Let* $\overline{y} \in X$ *be any point such that* $f_\varphi(\overline{y}) < f_\varphi(x)$. *Then, with* d *replaced by* $\overline{d} = \overline{y} - x$, *the inequality (5) holds for a finite integer* i.

(R) *Assume that* ∇f *is Lipschitz continuous on* X, *and let* Φ *be strongly monotone on* X. *Then, for all* ℓ *such that* $x + \ell d \in X$,

$$f(x+\ell d) - f(x) \le \ell\,(-m_\Phi + M_{\nabla f}/2\ell)\,\|d\|^2. \tag{6}$$

Hence, if $\ell \in (0, 2m_\Phi/M_{\nabla f})$, *then* $f(x+\ell d) < f(x)$.

Note that the step lengths resulting from the use of Rules A and R vary with Φ; choosing Φ such that the value of m_Φ is large enough, both rules accept unit step lengths.

In Table 1, we present some instances of the CA algorithm, together with their respective characterizing cost approximating mappings Φ^t; if the algorithm is known under different names in the unconstrained and constrained cases, we present them both, separated by a semicolon. In the table, a dot denotes the identity mapping on $\Re^n$, Q^t is a symmetric and positive definite matrix in $\Re^{n\times n}$, γ_t is a positive scalar, and $r^t : X \mapsto \Re^n$ is a continuous and strongly monotone mapping on X. (A much longer list is found in [58].)

Table 1. Instances of the cost approximation algorithm

Algorithm ($X = \Re^n$; $X \subset \Re^n$)	Choice of Φ^t
—; Frank–Wolfe	0
Steepest descent; Gradient projection	$(1/\gamma_t)\cdot$
Newton	$\nabla^2 f(x^t)\cdot$
Variable metric; Scaled gradient projection	$Q^t\cdot$
Proximal point	$\nabla f + (1/\gamma_t)\cdot$
Regularization	$\nabla f + (1/\gamma_t) r^t$

1.3. Scope and preview

We will demonstrate how the structure of [CDP] can be exploited in CA algorithms, decomposition being enabled by choosing the cost approximating mappings to be separable with respect to the partition (1) of $\Re^n$. The resulting independent subproblems can either be solved sequentially or in parallel; thus the algorithms can be tailored for different computer architectures, and facilitate a decomposition of [CDP] into a sequence of problems in much smaller dimensions.

A few words on the definition of $\mathcal{C}$: the partition of $\Re^n$ which is defined by (1) is not the only one possible—a set X_i may itself be a Cartesian product of convex sets. The number of parallel processors available dictates how many subproblems [VIP_{Φ_i}] that can be solved simultaneously, and it may therefore be necessary to group together some of the independent variables, thus creating a smaller number of (larger) components. (Such an action may also be of advantage since, with fewer components, the approximation defined by [VIP_{Φ_i}] is more accurate; in the sequential version of the algorithm, this may be used to reach a certain level of accuracy in less iterations.) The choice of $\mathcal{C}$ may also aid in the construction of a partition which makes the most efficient use of the computer. For example, when [VIP_Φ] is too large to be mapped in its entirety onto the computer available, the choice of $\mathcal{C}$ may be adapted to this situation by defining a decomposition of [VIP_Φ] such that each subproblem is of maximal size, thus creating as few sequential groups of subproblems as possible. The proper choice of $\mathcal{C}$ may thus depend on the size and structure of the problem to be solved as well as on the computer available. The results obtained will not change with the definition of $\mathcal{C}$, except when the cardinality of $\mathcal{C}$ is significant (see, e.g., Theorem 14).

A convergence analysis is made for three decomposition versions of the cost approximation algorithm. In the analysis, we seek to find the weakest possible assumptions under which convergence to a stationary point is guaranteed, under different choices of cost approximating mappings and accuracies of computations in the two main steps of the algorithm. In the *first* version, which includes as a special case the classical Gauss–Seidel coordinate descent algorithm in unconstrained optimization, the variable components are iterated upon sequentially. This computing model is suitable for sequential computers; it is, however, the natural algorithm model to consider also for parallel computers when the problem data is impossible to accommodate simultaneously. In this combination of a parallel and sequential algorithm the set $\mathcal{C}$ is defined such that each subproblem corresponding to an element $i \in \mathcal{C}$ can be efficiently solved in parallel; the corresponding variables are then updated before considering another element in $\mathcal{C}$. In the *second* version, which includes as a special case the classical Jacobi approach in unconstrained optimization, the independent subproblems are—at least conceptually—solved in parallel, followed by a synchronized updating step. This computer model is especially suitable when the subproblems are uniform in complexity. The *third* version introduces asynchronous parallel computations, as a means to avoid the synchronization penalty associated with the updating step in the second approach. By removing the synchronization step, variable component updates and solutions of the independent subproblems are made out of phase with each other. We consider a partially asynchronous computing model, where it is assumed that there is an upper bound on the interprocessor communication delays.

Presented in the order given, the three decomposition versions define algorithms with increasing degrees of decomposition, parallelism and asynchronism; this is clearly reflected in the convergence results, which show that the admissible step lengths in the updating steps are (essentially) inversely proportional to these degrees.

A majority of the existing convergence analyses of asynchronous computations are made through the interpretation of the algorithm as that of finding a fixed point of a (possibly point–to–set) mapping (e.g., [11, 72, 46, 73]); in our case, it would, for example, be possible to utilize the result of Lemma 1.a and 1.b in such an analysis. We will, however, make use of the *descent* properties of the CA algorithm, that is, Lemma 1.e.

Decomposition algorithms for [CDP] of these forms are not entirely new; another purpose of this paper is to unify and extend results in this direction obtained earlier by Cohen [16, 17], Bertsekas and Tsitsiklis [11], and Tseng [67, 68]. Overviews of parallel differentiable optimization are found in [41, 11, 51].

The rest of the paper is organized as follows. In Section 2, we analyze the sequential approach. In Sections 3 and 4, the synchronized and partially asynchronous parallel algorithms, respectively, are studied. Finally, in Section 5, we discuss some possible extensions.

2. Sequential cost approximation algorithms

2.1. The sequential algorithm

The *sequential cost approximation algorithm* proceeds as follows. Given an iterate x^t at iteration t, choose an index $i_t \in \mathcal{C}$ and a cost approximating mapping $\Phi^t_{i_t}$, and solve the problem of finding $y^t_{i_t} \in X_{i_t}$ such that $(i = i_t)$

$[\mathrm{VIP}_{\Phi^t_i}]$

$$[\Phi^t_i(y^t_i) + \nabla_i f(x^t) - \Phi^t_i(x^t_i)]^{\mathrm{T}}(y_i - y^t_i) \geq 0, \qquad \forall y_i \in X_i.$$

(The case of Φ^t_i being a gradient corresponds to solving a convex problem of the form $[\mathrm{CDP}_{\varphi^t_i}]$.) We then let $y^t_j = x^t_j$ for all $j \in \mathcal{C} \setminus \{i_t\}$ and $d^t = y^t - x^t$. The next iterate, x^{t+1}, then is defined by $x^{t+1} = x^t + \ell_t d^t$, that is,

$$x^{t+1}_j = \begin{cases} x^t_j + \ell_t(y^t_j - x^t_j), & j = i_t, \\ x^t_j, & j \neq i_t, \end{cases}$$

for some value of ℓ_t such that $x^t_{i_t} + \ell_t(y^t_{i_t} - x^t_{i_t}) \in X_{i_t}$ and the value of f is reduced sufficiently.
A summary of the algorithm is given in Table 2. (The choices of the sequences $\{\Phi^t\}$ and $\{i_t\}$ are not explicitly stated; they are either chosen *a priori* or adaptively.)

Table 2. The sequential cost approximation algorithm

0. (*Initialization*): Choose an initial point $x^0 \in X$, and let $t = 0$.
1. (*Search direction generation*): Find a solution $y^t_{i_t}$ to $[\mathrm{VIP}_{\Phi^t_{i_t}}]$. Let $y^t_j = x^t_j$ for all $j \neq i_t$. The resulting search direction is $d^t = y^t - x^t$.
2. (*Termination criterion*): If x^t_i solves $[\mathrm{VIP}_{\Phi^t_i}]$ for all $i \in \mathcal{C} \to$ Stop ($x^t \in \Omega$). Otherwise, continue.
3. (*Line search*): Choose a step length, ℓ_t, such that $x^t_{i_t} + \ell_t d^t_{i_t} \in X_{i_t}$ and the value of f is reduced sufficiently.
4. (*Update*): Let $x^{t+1} = x^t + \ell_t d^t$, and $t := t + 1$.
5. (*Termination criterion*): If x^t is acceptable as a solution $\to$ Stop. Otherwise, go to Step 1.

In the above algorithm, let the sequence $\{i_t\}$ be chosen according to the *cyclic rule*, that is,

$$i_t = t \ (\mathrm{mod}\ |\mathcal{C}|) + 1. \tag{7}$$

Choose the cost approximating mapping

$$\Phi^t_i = \nabla_i f(x^t_{i_-}, \cdot, x^t_{i_+}). \tag{8}$$

We note that this mapping is monotone whenever f is convex in x_i. Since $\Phi^t_i(x^t_i) = \nabla_i f(x^t)$, the subproblem $[\mathrm{VIP}_{\Phi^t_i}]$ is equivalent (under this convexity assumption) to finding

$$y^t_i \in \arg\min_{y_i \in X_i} f(x^t_{i_-}, y_i, x^t_{i_+}).$$

An exact line search would produce $\ell_t = 1$, since y^t_i minimizes f over X_i (the remaining components of x kept fixed), and so $x^{t+1}_i = y^t_i$. The iteration described is that of the classical *Gauss–Seidel* algorithm [50] (also known as the *relaxation algorithm*, the *coordinate descent* *method*, and the *method of successive displacements*), originally proposed for the solution of unconstrained problems. The Gauss–Seidel algorithm is hence a special case of the sequential CA algorithm given by the choices of the cyclic index ordering, the cost approximating mapping (8) and an exact line search.

In what follows, we shall extend the Gauss–Seidel algorithm in several directions. We will establish convergence for more general cost approximating mappings and index orderings, and for inexact solutions of the subproblems and line searches.

2.2. Convergence of the conceptual algorithm

In our first convergence result, we shall assume that the sequence $\{\Phi^t\}$ of mappings is given by a monotone gradient mapping on $X \times X$. (A non-gradient mapping Φ does not necessarily yield a descent direction if it is monotone only.) We thus introduce the function $\varphi : X \times X \mapsto \Re$ defined by

$$\varphi(y, x) = \sum_{i \in \mathcal{C}} \varphi_i(y_i, x), \qquad x, y \in X, \tag{9}$$

where $\varphi_i : X_i \times X \mapsto \Re$ is continuous on $X_i \times X$ and convex and in C^1 on X_i with respect to y_i. [Hence, $\Phi_i^t \equiv \nabla_{y_i} \varphi(\cdot, x^t)$.] Further, we shall assume that the line search is exact.

THEOREM 1 (Convergence of the cyclic CA algorithm under Rule E) *Let the sequence $\{\Phi^t\}$ of cost approximating mappings be given by the gradient of the convex function φ in (9). In the sequential CA algorithm, let the sequence $\{i_t\}$ be chosen according to the cyclic rule (7), and Rule E be used. Assume that $x^0 \in X$ is such that the lower level set $L(x^0)$ is bounded, and that the problem* [CDP$_{\varphi_i}$] *is well defined, in the sense that the set $Y_i(x)$ is nonempty and bounded for all $x \in L(x^0)$. Then, $\{f(x^t)\} \to f(\overline{x})$ for some $\overline{x} \in \Omega$, any accumulation point of the sequence $\{x^t\}$ (at least one such point exists) lies in Ω, and*

$$\left\{ \inf_{x \in \Omega \cap L(x^0)} \|x^t - x\| \right\} \to 0. \tag{10}$$

Proof: To prove this result, we shall apply Zangwill's Theorem [76, Sec. 4.5]. We first identify the *solution set* with Ω, the continuous *descent function* with f, and the *algorithmic map* with the composite map $A = L \circ D_{|\mathcal{C}|} \circ L \circ D_{|\mathcal{C}|-1} \circ \ldots \circ L \circ D_1$, where $x \mapsto D_i(x) = Y_i(x) - x_i$, $i \in \mathcal{C}$, is the search direction finding map, and L is the exact line search map. To fulfill the assumptions of Zangwill's Theorem, we need to show that (i) [compactness] the sequence $\{x^t\}$ lies in a compact set, (ii) [adaption] if $x^t \notin \Omega$ then $f(x^{t+1}) < f(x^t)$, and if $x^t \in \Omega$ then either the algorithm terminates or $f(x^{t+1}) \geq f(x^t)$, and (iii) [closedness] the algorithmic map A is closed at all points in $L(x^0) \setminus \Omega$. (Observe that the algorithmic map describes a full cycle of $|\mathcal{C}|$ iterations in the algorithm described in Table 2.)

(i) The lower level set $L(x^0)$ is closed (since f is continuous) and, by assumption, therefore compact. The descent property (ii) ensures that $\{x^t\}$ lies in this set.

(ii) If $x^t \notin \Omega$, then Lemma 1.e.ii establishes that d^t defines a direction of descent with respect to f. The descent property then follows from the use of Rule E. If $x^t \in \Omega$, then by the fixed point property of the mapping $x \mapsto Y_i(x)$, $i \in \mathcal{C}$ (Lemma 1.a.ii), it follows that $x_i^t \in Y_i(x^t)$ for all $i \in \mathcal{C}$, in which case the algorithm is terminated at Step 2.

(iii) Let $i \in \mathcal{C}$. The set $Y_i(x)$ is characterized by

$$Y_i(x) = \partial([\varphi_i(\cdot, x) + \delta_{X_i}]^\circ)(\nabla_{y_i} \varphi(x_i, x) - \nabla_i f(x)), \qquad x \in X \tag{11}$$

(cf. Lemma 1.b). The assumption that $Y_i(x)$ is nonempty and bounded for all $x \in L(x^0)$ is equivalent to

$$\nabla_{y_i}\varphi(x_i, x) - \nabla_i f(x) \in \mathtt{int}\,(\mathtt{dom}\,([\varphi_i(\cdot, x) + \delta_{X_i}]^\circ)), \qquad x \in L(x^0) \tag{12}$$

(cf. Lemma 1.c.ii).

The function $[\varphi_i(\cdot, x)+\delta_{X_i}]^\circ$ is closed, proper and convex ([74, Sec. 5.4] and [62, Thm. 12.2]), and $\mathtt{int}\,(\mathtt{dom}\,([\varphi_i(\cdot, x)+\delta_{X_i}]^\circ)) = \mathtt{int}\,(\mathtt{dom}\,\partial([\varphi_i(\cdot, x)+\delta_{X_i}]^\circ))$ [62, Thm. 23.4]. The mapping $\partial([\varphi_i(\cdot, x) + \delta_{X_i}]^\circ)$ is monotone, and therefore, by (12) and Proposition 32.33 of [77], locally bounded at $\nabla_{y_i}\varphi(x_i, x) - \nabla_i f(x)$. From (11), the mapping Y_i is therefore locally bounded on $L(x^0)$. From (i), the sequence $\{x^t\}$ is bounded. The local boundedness of Y_i, $i \in \mathcal{C}$, implies that also the sequence $\{y^t\}$ is bounded. This sequence then has an accumulation point. The map D_i is closed on X by Lemma 1.d. Since the map L is closed on $L(x^0)$, Lemma 4.2 of [76] then ensures that the composite map $L \circ D_i$ is closed on $L(x^0)$. It is also compact valued, from the descent property and the compactness of $L(x^0)$. Therefore, appealing to Corollary 4.2.1 of [76], the map A is closed on $L(x^0)$.

The assumptions of Zangwill's Theorem are thus fulfilled, and we conclude that $\{f(x^t)\}$ converges to $f(\overline{x})$ for some $\overline{x} \in \Omega$, and any accumulation point of $\{x^t\}$ lies in Ω. (The existence of at least one such point follows from the boundedness of $\{x^t\}$.) By the compactness of $L(x^0)$, the property (10) follows from Theorem 14.1.4 of [50]. ■

Letting $|\mathcal{C}| = 1$, the cyclic CA algorithm reduces to the original algorithm [54, Chap. 4], and the convergence conditions are identical to those of the original one.

With the choice (8) of Φ, the Gauss–Seidel algorithm is obtained. For this algorithm, the sequences $\{y^t\}$ and $\{x^{t+1}\}$ coincide, and thus, from the remaining assumptions of the theorem, the sequence $\{y^t\}$ is automatically bounded. For the unconstrained case, a convergence result corresponding to that of the above theorem is given by Zadeh [75]; see also [11, Prop. 3.2.5] and [50, Thm. 14.6.7].

A generalization of Zangwill's classical result will enable us to weaken the assumption that the subproblems [CDP_{φ_i}] are solved exactly, and also allows for the use of more general rules for choosing indices. The proof technique used is based on the *Spacer Step Theorem* [42, Sec. 7.10]; the idea behind the spacer step is that in the sequence $\{x^t\}$ generated by an algorithm one identifies an infinite subsequence whose generation can be described by an algorithmic map A satisfying the conditions of Zangwill's Theorem. Under the assumption that the remaining part of the sequence is such that the value of f never increases, convergence is guaranteed from that of the algorithm A; this is easily established by letting the remaining part of the complete algorithm be described by the mapping

$$C(x) = \{\, y \in X \mid f(y) \leq f(x) \,\}, \qquad x \in X; \tag{13}$$

clearly, C is closed on X, and the complete algorithmic map, $C \circ A$, is also closed on X whenever $L(x^0)$ is bounded.

2.3. Truncated cyclic cost approximation

Although each subproblem [CDP_{φ_i}] may be of much smaller dimension than the original problem, it can still be computationally demanding to solve, and in practice it may not be possible to solve it exactly. Motivated by these practical considerations, we will introduce a truncated version of the sequential CA algorithm in which the solution of [CDP_{φ_i}] is terminated before reaching its optimal solution. The basis for the algorithm described and validated in this section is Lemma 1.e.ii. (Truncated CA algorithms can also be based on Lemma 1.e.iii.) We will assume that the algorithm employed for solving [CDP$_{\varphi_i}$], and the termination criteria used for it, satisfy the following conditions.

ASSUMPTION 1 (Properties of a truncated algorithm) *Given an* $x \in X$, *the problem* [CDP$_{\varphi_i}$] *is solved, starting from* x_i, *with an iterative algorithm where one iteration may be described by the algorithmic map* $B_i : X_i \times X \mapsto 2^{X_i} \times X$ *with the following properties.*

(i) (Fixed point) $z_i \in Y_i(x) \Longleftrightarrow (z_i, x) \in B_i(z_i, x)$, *in which case the algorithm terminates.*

(ii) (Descent) *If* $x \notin \Omega$ *and* $z_i \notin Y_i(x)$, *then* $f_{\varphi_i}(y_i) < f_{\varphi_i}(z_i)$ *for all* y_i *with* $(y_i, x) \in B_i(z_i, x)$.

(iii) (Closedness) *The map* B_i *is u.s.c., closed and compact valued on* $X_i \times L(x^0)$.

(iv) (Finite termination) *The number of iterations performed is bounded from below by 1, and from above by a positive integer* $\overline{k}_i$.

The condition (i) ensures that the fixed point property of the original algorithm is preserved, and that the subproblem algorithm terminates whenever a solution to [CDP$_{\varphi_i}$] has been obtained. The condition (ii), together with the lower bound in (iv) on the number of iterations performed and Lemma 1.e.ii, ensures that the resulting approximate solution $\overline{y}_i$ of [CDP$_{\varphi_i}$] defines a direction of descent with respect to f. Condition (iii) ensures the closedness of the map describing the overall algorithm. (Whenever X is bounded, it can be replaced by the assumption that the map B_i is closed on $X_i \times X$.) Condition (iv), finally, limits the work done in each main iteration of the cyclic CA algorithm. If X is bounded, then the map B_i describing *any* truncated CA algorithm based on a function $\hat{\varphi}_i : X_i \times X_i \mapsto \Re$ of the form $\hat{\varphi}_i(y_i, x_i)$ together with *any* closed updating step which guarantees descent (such as Rule E or a fixed relaxation step) satisfies Assumption 1.i–1.iii.

The truncated cyclic CA algorithm is obtained from replacing Step 1 of the cyclic CA algorithm (see Table 2) by:

1'. (*Search direction generation*): Apply $1 \le k_{i_t}^t \le \overline{k}_i$ iterations of the algorithm described by B_{i_t}, starting from $x_{i_t}^t$, that is, let

$$(\overline{y}_{i_t}^t, x^t) \in \underbrace{B_{i_t} \circ \ldots \circ B_{i_t}}_{k_{i_t}^t \text{ times } B_{i_t}}(x_i^t, x^t).$$

Let $\overline{y}_j^t = x_j^t$ for all $j \neq i_t$. The resulting search direction is $d^t = \overline{y}^t - x^t$.

Note that the value of $k_{i_t}^t$ need not be determined *a priori*, and could instead be regarded as a product of the algorithm and the termination criteria chosen. (We only require that the upper bound exists.)

THEOREM 2 (Convergence of the truncated cyclic CA algorithm) *Replace Step 1 by Step 1' in the cyclic CA algorithm. In Theorem 1, replace the condition that the set* $Y_i(x)$ *is nonempty and bounded for all* $x \in L(x^0)$ *with Assumption 1. Then, the conclusions of Theorem 1 hold for the truncated cyclic CA algorithm.*

Proof: We begin by identifying the algorithmic map A. We assume that $\{x^t\}$ is infinite (otherwise, by Assumption 1.i, with $z_i = x_i$, the algorithm is terminated at a point in Ω). Let $i \in \mathcal{C}$ and $\mathcal{T}^i$ be the subset of $\mathcal{Z}_+$ in which the index i is chosen. Assumption 1.iv and the use of the cyclic rule imply that $\{k_i^t\}_{\mathcal{T}^i}$ is an infinite, bounded sequence of positive integers. There must then be at least one positive integer, say k_i, that occurs an infinite number of times in the sequence $\{k_i^t\}_{\mathcal{T}^i}$. We then define

$$\overline{Y}_i(x) = \{\, \overline{y}_i \in \Re^{n_i} \mid (\overline{y}_i, x) \in \underbrace{B_i \circ \ldots \circ B_i}_{k_i \text{ times } B_i}(x_i, x) \,\}, \qquad i \in \mathcal{C}.$$

The algorithmic map is $A = E \circ \overline{D}_{|\mathcal{C}|} \circ E \circ \overline{D}_{|\mathcal{C}|-1} \circ \ldots \circ E \circ \overline{D}_1$, where $\overline{D}_i(x) = \overline{Y}_i(x) - x_i$.

We next establish that this map has the properties desired (boundedness, adaption, closedness). We first note that Assumption 1.ii, together with Lemma 1.e.ii, yields that the map $\overline{D}_i$, $i \in \mathcal{C}$, has the descent property. From the use of Rule E, it then follows that $f(\overline{x}) < f(x)$ for any $\overline{x} \in L \circ \overline{D}_i(x)$ and all $i \in \mathcal{C}$, and therefore A satisfies the second condition of Zangwill's Theorem (adaption). The boundedness assumption on the lower level set $L(x^0)$ then implies that the sequence $\{x^t\}$ is bounded, thereby satisfying the first condition of Zangwill (boundedness).

To establish the third condition (closedness), we consider an $i \in \mathcal{C}$. The map B_i is closed on $X_i \times L(x^0)$ (Assumption 1.iii). By the upper semicontinuity and compactness of B_i (Assumption 1.iii again), Lemma 12 of [44] implies that if $\overline{y}_i^\infty$ is an accumulation point of a sequence $\{\overline{y}_i^t\}$ given by $(\overline{y}_i^{t+1}, x) \in B_i(\overline{y}_i^t, x)$, then the set $\{\, y_i \mid (y_i, x) \in B_i \circ B_i(\overline{y}_i^\infty, x) \,\}$ contains an accumulation point of $\{\overline{y}_i^t\}$. We are then in the position of using Lemma 4.2 of [76] to conclude that the composite map $B_i \circ B_i$ is closed on $X_i \times L(x^0)$. This result used repeatedly then establishes that the map $\overline{Y}_i$, and therefore also $\overline{D}_i$, is closed on $L(x^0)$. Applying Lemma 4.2 of [76] again, we may conclude that the map $L \circ \overline{D}_i$ is closed. Moreover, the boundedness of $L(x^0)$ and the descent property of $L \circ \overline{D}_i$ ensures that the range of this map over $L(x^0)$ is a compact set. We invoke Corollary 4.2.1 of [76] to conclude that the algorithmic map A is closed on $L(x^0)$.

We identify the map C by, for all $i \in \mathcal{C}$, extracting the value k_i from the possible choices in the sequence $\{k_i^t\}_{\mathcal{T}^i}$; this map is clearly of the form (13). Hence, all the conditions of the Spacer Step Theorem [42, p. 231] are fulfilled. The rest of the proof follows that of Theorem 1. ∎

2.4. Essentially cyclic cost approximation

We next replace the cyclic ordering by one which allows for a large freedom in selecting indices. In the *essentially* (or, *almost*) *cyclic rule* (e.g., [25]) it is assumed that there exists a positive integer $N \geq |\mathcal{C}|$ such that every index in $\mathcal{C}$ is chosen at least once every N successive iterations:

$$\mathcal{C} \subseteq \{i_t, i_{t+1}, \ldots, i_{t+N-1}\}, \qquad t = 1, 2, \ldots. \tag{14}$$

Within the limits defined by the value of N, one may use any strategy for choosing indices, and (14) is therefore not a rule *per se.* (With $N = |\mathcal{C}|$, it however reduces to the cyclic rule.) An example of a reasonable ordering rule is the choice of the index which corresponds to the variable components being farthest (in some measure) from being optimal; the Gauss–Southwell (remotest) order [12] is such an example.

In analyzing this algorithm, we shall again make use of the Spacer Step Theorem with a well-defined choice of A and C.

THEOREM 3 (Convergence of the essentially cyclic CA algorithm) *Replace the cyclic rule (7) by the essentially cyclic rule (14) in the cyclic CA algorithm. Then, the conclusions of Theorem 1 holds for the essentially cyclic CA algorithm.*

Proof: We assume, as before, that the sequence $\{x^t\}$ is infinite. Since the value of N is finite, in the sequence $\{i_t\}$ of indices chosen there is at least one consecutive sequence $(i_t, i_{t+1}, \ldots, i_{t+N-1})$ of N elements of $\mathcal{C}$, say $(\imath_1, \imath_2, \ldots, \imath_N)$, that occurs an infinite number of times. Let $A = E \circ D_{\imath_N} \circ E \circ D_{\imath_{N-1}} \circ \ldots \circ E \circ D_{\imath_1}$. From the proof of Theorem 1, since $\mathcal{C} \subseteq \{\imath_1, \imath_2, \ldots, \imath_N\}$ [cf. (14)], the map A has all the properties required to invoke Zangwill's Theorem.

The remaining part of the algorithm (the consecutive sequence $(\imath_1, \imath_2, \ldots, \imath_N)$ removed from the possible choices of indices in $\{i_t\}$) is clearly of the form (13). We can therefore invoke the Spacer Step Theorem, and the rest of the proof follows that of Theorem 1. ■

In the context of the traffic assignment problem (see Section 1.1), the commodity $i_t = (p, q)$ may be chosen based on the shortest route travel costs, given the link flows x^t (e.g., [59, 47, 53, 56]).

Combining the use of the essentially cyclic rule and the truncated subproblem algorithm into a *truncated essentially cyclic CA algorithm* is validated next.

THEOREM 4 (Convergence of the truncated essentially cyclic CA algorithm) *Replace the cyclic rule (7) by the essentially cyclic rule (14) in the truncated cyclic CA algorithm. Then the conclusions of Theorem 2 holds for the truncated essentially cyclic CA algorithm.*

Proof: Let $A = L \circ \overline{D}_{\imath_N} \circ L \circ \overline{D}_{\imath_{N-1}} \circ \ldots \circ L \circ \overline{D}_{\imath_1}$, where the sequence $(\imath_1, \imath_2, \ldots, \imath_N)$ is defined as in the proof of Theorem 3. Since $\mathcal{C} \subseteq \{\imath_1, \imath_2, \ldots, \imath_N\}$ (cf. the essentially cyclic rule), using exactly the same arguments as in the proof of Theorem 2, the result follows. ■

We next replace the exact line search by the Armijo step length rule. (Since the Armijo rule is here used on an individual component x_i only, in (5), different values of α and β may be chosen for the different components.)

The sequential CA algorithm using Rule A will be established (under the essentially cyclic rule) for strictly monotone mappings Φ of the form

$$\Phi(y, x)^{\mathrm{T}} = [\Phi_1(y_1, x)^{\mathrm{T}}, \ldots, \Phi_{|\mathcal{C}|}(y_{|\mathcal{C}|}, x)^{\mathrm{T}}], \qquad x, y \in X; \tag{15}$$

to ensure that the subproblem solution is bounded, we also impose the condition corresponding to (3), that is,

$$\Phi_i(x_i, x) - \nabla_i f(x) \in \mathtt{int}\,(\mathtt{dom}\,([\Phi_i(\cdot, x) + N_{X_i}]^{-1})), \qquad x \in L(x^0). \tag{16}$$

THEOREM 5 (Convergence of the essentially cyclic CA algorithm under Rule A) *Let the sequence $\{\Phi^t\}$ of cost approximating mappings be given by a continuous mapping on $X \times X$ of the form (15), where $\Phi_i(\cdot, x)$ is maximal monotone and strictly monotone on X_i. In the sequential CA algorithm, let the sequence $\{i_t\}$ be chosen according to the essentially cyclic rule (14), and Rule A be used. Assume that $x^0 \in X$ is such that the lower level set $L(x^0)$ is bounded, and that the problem $[\mathrm{VIP}_{\Phi_i}]$ is well defined, in the sense that (16) holds. Then, $\{f(x^t)\} \to f(\overline{x})$ for some $\overline{x} \in \Omega$, any accumulation point of the sequence $\{x^t\}$ (at least one such point exists) lies in Ω, and (10) holds.*

Proof: Let x^∞ be an accumulation point of the sequence $\{x^t\}$, corresponding to a convergent subsequence $\mathcal{T}$; the existence of such a point follows from the boundedness of $L(x^0)$ and the descent property. To establish that also the sequence $\{y^t\}$ is bounded, we first note that the mapping $[\Phi_i(\cdot, x) + N_{X_i}]^{-1}$, $i \in \mathcal{C}$, is maximal monotone ([63, Thm. 3] and [77, Prop. 32.5]). Then, (16) implies, by Theorem 1 of [61], that it is locally bounded at $\Phi_i(x_i, x) - \nabla_i f(x)$. By the characterization $Y_i(x) = [\Phi_i(\cdot, x) + N_{X_i}]^{-1}(\Phi_i(x_i, x) - \nabla_i f(x))$, $x \in X$, of the set $Y_i(x)$ (see Lemma 1.b) and (16), the map Y_i is locally bounded on $L(x^0)$. Since the sequence $\{x^t\}$ is bounded, it follows that also the sequence $\{y^t\}$ is bounded. Let y^∞ be an arbitrary accumulation point of $\{y^t\}$, which we, for simplicity, will also associate with the convergent subsequence $\mathcal{T}$.

Hence, the sequence $\{d^t\}$ is bounded. We next show that

$$\{\nabla f(x^t)^{\mathrm{T}} d^t\} \to 0. \tag{17}$$

If $\{d^t\} \to 0$, then (17) follows from the boundedness of $\{\nabla f(x^t)\}$. Assume therefore that $\{d^t\} \not\to 0$. By Lemma 2.A.i, the characterization [VIP$_\Phi$] of y^t, and the strict monotonicity of $\Phi_i(\cdot, x^t)$,

$$f(x^{t+1}) - f(x^t) \leq \alpha\ell_t \nabla f(x^t)^{\mathrm{T}} d^t \leq \alpha\ell_t [\Phi(x^t, x^t) - \Phi(y^t, x^t)]^{\mathrm{T}} d^t < 0, \qquad t = 0, 1, \ldots.$$

If $\{d^t\} \not\to 0$, then as $\{f(x^{t+1}) - f(x^t)\} \to 0$ holds, there must be a subsequence $\overline{\mathcal{T}}$ such that $\{\ell_t\}_{\overline{\mathcal{T}}} \to 0$. There must then be an index $\bar{t}$ such that for every $t \geq \bar{t}$ in $\overline{\mathcal{T}}$, the step length produced by Rule A is less than one, that is,

$$f(x^t + (\ell_t/\beta)d^t) - f(x^t) > \alpha(\ell_t/\beta)\nabla f(x^t)^{\mathrm{T}} d^t, \qquad t \geq \bar{t}, \qquad t \in \overline{\mathcal{T}}. \tag{18}$$

Dividing both sides of this inequality by ℓ_t/β, in the limit of $\overline{\mathcal{T}}$ of the resulting inequality we reach the conclusion that $\nabla f(x^\infty)^{\mathrm{T}} d^\infty \geq 0$, where $d^\infty = y^\infty - x^\infty$, while in the limit of the descent inequality $\nabla f(x^t)^{\mathrm{T}} d^t < 0$, the reverse inequality is obtained. The accumulations points were chosen arbitrarily, whence (17) follows.

Let $i \in \mathcal{C}$. If i does not occur an infinite number of times in $\{i_t\}_{t \in \mathcal{T}}$, then by construction $y_i^t = x_i^t$ holds for all sufficiently large t, and hence $y_i^\infty = x_i^\infty$ holds. Assume that it does occur an infinite number of times, and that $y_i^\infty \neq x_i^\infty$. In the limit of $\mathcal{T}$ of [VIP$_{\Phi_i^t}$], using the strict monotonicity of Φ_i we then reach a contradiction to (17). Thus, $y^\infty = x^\infty$ must actually hold.

From $\mathcal{T}$ we extract a subsequence, $\overline{\mathcal{T}}$, where we assume that $(i_t, i_{t+1}, \ldots, i_{t+N-1})$ is the same for all t, say $(\imath_0, \imath_1, \ldots, \imath_{N-1})$ [cf. the proof of Theorem 3]. From the use of the Armijo rule, $\|x^{t+1} - x^t\| = \beta^{\bar{\imath}_t}\|d^t\| \leq \|d^t\|$ holds, and since $\{d^t\}_{t \in \overline{\mathcal{T}}} = \{y^t - x^t\}_{t \in \overline{\mathcal{T}}} \to 0$, $\{x^{t+1} - x^t\}_{t \in \overline{\mathcal{T}}} \to 0$ holds, that is, $\{x^{t+1}\}_{t \in \overline{\mathcal{T}}} \to x^\infty$ follows. Repeated use of this argument shows that $\{x^{t+j}\}_{t \in \overline{\mathcal{T}}} \to x^\infty$ (and $\{y^{t+j}\}_{t \in \overline{\mathcal{T}}} \to x^\infty$) for $j = 0, 1, \ldots, N-1$. Since

$$[\Phi_{\imath_j}(y_{\imath_j}^{t+j}, x^{k+j}) + \nabla_{\imath_j} f(x^{t+j}) - \Phi_{\imath_j}(x_{\imath_j}^{t+j}, x^{t+j})]^{\mathrm{T}}(y_{\imath_j} - y_{\imath_j}^{t+j}) \geq 0, \qquad \forall y_{\imath_j} \in X_{\imath_j}$$

holds for $j = 0, 1, \ldots, N-1$ and $t \in \overline{\mathcal{T}}$, using the above result yields that

$$\nabla_{\imath_j} f(x^\infty)^{\mathrm{T}}(x_{\imath_j} - x_{\imath_j}^\infty) \geq 0, \qquad \forall x_{\imath_j} \in X_{\imath_j}, \qquad j = 0, 1, \ldots, N-1. \tag{19}$$

Since $\mathcal{C} \subseteq \{\imath_0, \imath_1, \ldots, \imath_{N-1}\}$, it follows that

$$\nabla_i f(x^\infty)^{\mathrm{T}}(x_i - x_i^\infty) \geq 0, \qquad \forall x_i \in X_i, \qquad i \in \mathcal{C}.$$

Adding these inequalities over $i \in \mathcal{C}$, we obtain that $x^\infty \in \Omega$. The final statement follows as in the proof of Theorem 1. ■

Tracing the proof, we may conclude that with *any* rule for choosing indices,

$$\nabla_i f(x^\infty)^{\mathrm{T}}(x_i - x_i^\infty) \geq 0, \qquad \forall x_i \in X_i$$

holds for those indices i which occur infinitely many times in building the convergent subsequence $\{x^t\}_{\mathcal{T}}$. (This is true assuming only monotonicity of $\Phi_i(\cdot, x)$ provided that it is a gradient mapping.) To ensure that it holds for the remaining indices, (14) is however necessary. This result complies with those obtained for fixed point algorithms using the *free-steering* order (e.g., [50, 25]), where it is assumed that each index is chosen an infinite number of times in $\{i_t\}$.

Strict monotonicity of $\Phi_i(\cdot, x)$ is essential for convergence in the sequential algorithm. In the original algorithm ($|\mathcal{C}| = 1$), monotonicity suffices if $\Phi_i(\cdot, x)$ is a gradient mapping; this is therefore also true for the synchronized parallel algorithm (see Theorem 10). Also, in the original, conceptual algorithm as well as in the special case of the synchronized parallel algorithm, it is possible to establish convergence where Rule A is combined with the truncated subproblem algorithm (see Theorem 11). We have not been able to extend that result to the case of sequential decomposition, since it is not guaranteed that $\{d^t\} \to 0$ holds.

Tseng [67] establishes a result similar to Theorem 5 for the nonlinear proximal descent algorithm (an instance of the CA algorithm; see [53] and [54, Thm. 4.1]); the boundedness of $\{x^t\}$ and $\{y^t\}$ is there ensured by a boundedness assumption on X.

Below, we establish convergence for the essentially cyclic CA algorithm for strongly monotone cost approximating mappings, under the three step length rules E, A and R. We introduce the notation ℓ_t^i for the step length taken in iteration t, where i is the index chosen, and $\overline{\ell}_t^i$ is the largest step length ℓ^i in the direction of d_i^t for which $x_i^t + \ell^i d_i^t \in X_i$.

THEOREM 6 (Convergence of the essentially cyclic CA algorithm under strongly monotone mappings Φ^t) *Assume that f is bounded from below on X. For each $i \in \mathcal{C}$ and t, let Φ_i^t be strongly monotone on X_i and Lipschitz continuous on X_i. Let further $m_{\Phi_i} := \liminf_{t\to\infty}\{m_{\Phi_i^t}\} > 0$ and $M_{\Phi_i} := \limsup_{t\to\infty}\{M_{\Phi_i^t}\} < +\infty$ for all $i \in \mathcal{C}$. In the sequential CA algorithm, let the sequence $\{i_t\}$ be chosen according to the essentially cyclic rule (14). Then, from any starting point $x^0 \in X$, under Rule A, any accumulation point of $\{x^t\}$ lies in Ω. The same conclusion holds for Rule E whenever $\{\ell_t^i\}$, $i \in \mathcal{C}$, is bounded from above.*

Assume further that $\nabla_i f$ is Lipschitz continuous on X_i for all $i \in \mathcal{C}$. If, in Rule R,

$$\ell_t^i \in \left[\varepsilon^i, \min\left\{\overline{\ell}_t^i, 2m_{\Phi_i^t}/M_{\nabla_i f} - \varepsilon^i\right\}\right], \qquad \varepsilon^i \in (0, m_{\Phi_i}/M_{\nabla_i f}], \qquad i \in \mathcal{C},$$

then the same conclusion holds for Rule R.

If further $\{x^t\}$ is bounded and Ω is nonempty, then for the Rules E, A and R, (10) holds.

If further Ω is finite, then for the Rules E, A and R, $\{x^t\}$ converges to a point in Ω.

Proof: Similarly to the proof of Theorem 5 for Rule A, utilizing Lemmas 1.e.iii and 2.A.i,

$$f(x^{t+1}) - f(x^t) \le \alpha \ell_t \nabla f(x^t)^{\mathrm{T}} d^t \le -\alpha \ell_t m_{\Phi_{i_t}^t} \|d^t\|^2, \qquad t = 0, 1, \ldots \tag{20}$$

holds. Since f is lower bounded, it follows as in that proof that $\{\nabla f(x^t)^{\mathrm{T}} d^t\} \to 0$, and further that $\{d^t\} \to 0$. Since the left–hand–side of the first inequality in (20) is minimized in Rule E, it is clear that the same holds for this rule. For Rule R, finally, we utilize Lemma 2.R to reach the inequality

$$f(x^t + \ell^i d^t) - f(x^t) \le \ell^i \left(-m_{\Phi_i^t} + M_{\nabla_i f}/2\ell^i\right) \|d^t\|^2, \qquad \ell^i \in [0, \overline{\ell}^i].$$

From this inequality, it is clear that

$$f(x^{t+1}) - f(x^t) \le -\rho \|d^t\|^2 \tag{21}$$

holds, with $\rho = \varepsilon^2 M_{\nabla_i f}/2$. Since f is lower bounded on X, (21) implies that $\{d^t\} \to 0$ holds for this step length rule as well.

Assume that $\{x^t\}$ has an accumulation point, x^∞, corresponding to a convergent subsequence $\mathcal{T}$. We then have that $\{y^t\}_{t\in\mathcal{T}} \to x^\infty$. In each of the step length rules, ℓ^i is upper bounded; hence, for some $\delta > 0$, $\|x^{t+1} - x^t\| \le \delta \|d^t\|$ holds. Constructing a convergent subsequence $\overline{\mathcal{T}}$ in a manner similar to that in the proof of Theorem 5, we may then establish that $\{x^{t+j}\}_{t\in\overline{\mathcal{T}}} \to x^\infty$ and $\{y^{t+j}\}_{t\in\overline{\mathcal{T}}} \to x^\infty$, $j = 0, 1, \ldots, N-1$. The points $y_{i_j}^{t+j}$, $j = 0, 1, \ldots, N-1$, are given by

$$[\Phi_{i_j}^{t+j}(y_{i_j}^{t+j}) + \nabla_{i_j} f(x^{t+j}) - \Phi_{i_j}^{t+j}(x_{i_j}^{t+j})]^{\mathrm{T}}(y_{i_j} - y_{i_j}^{t+j}) \ge 0, \qquad \forall y_{i_j} \in X_{i_j}.$$

The Lipschitz continuity of $\Phi_{i_j}^{t+j}$ then yields that

$$\nabla_{i_j} f(x^{t+j})^{\mathrm{T}}(y_{i_j} - y_{i_j}^{t+j}) \geq -M_{\Phi_{i_j}} \|y_{i_j}^{t+j} - x_{i_j}^{t+j}\| \cdot \|y_{i_j} - y_{i_j}^{t+j}\|, \qquad \forall y_{i_j} \in X_{i_j},$$

for $j = 0, 1, \ldots, N-1$. In the limit of $t \in \overline{\mathcal{T}}$, this inequality implies that (19) holds for $j = 0, 1, \ldots, N-1$, that is, $x^\infty \in \Omega$. The result (10) follows as in the proof of Theorem 1. The last result follows from a known result on bounded sequences $\{x^t\}$ satisfying $\{x^{t+1} - x^t\} \to 0$ [50, Thm. 14.1.5]. ■

Remark. In Rule E, the upper bound on ℓ^i may be any positive scalar; whenever $L(x^0)$ is bounded, the assumption is superfluous.

Tracing the proof, convergence is guaranteed for *any* step length rule that results in a larger reduction of f at each iteration than the Armijo rule.

If the lower boundedness of f is replaced by the stronger condition of *weak coercivity* of f on X ($\{f(x)\} \to +\infty$ when $\{\|x\|\} \to +\infty$ and $x \in X$), then the sequences $\{x^t\}$ and $\{y^t\}$ are guaranteed to be bounded.

To ensure that a unit step implies convergence in Rule R, the cost approximating mapping Φ_i should be chosen such that [cf. (2.4)]

$$\frac{2m_{\Phi_i}}{M_{\nabla_i f}} > 1, \qquad i \in \mathcal{C}. \tag{22}$$

(The Armijo rule accepts a unit step size under the similar condition that $2(1-\alpha)m_{\Phi_i}/M_{\nabla_i f} \geq 1$.)

Cohen [16, 17] establishes a result similar to the above theorem for the auxiliary problem principle (an instance of the CA algorithm [54, Th 4.1]) in the case of Rule R; he, however, further assumes that f is strongly convex in each component x_i, and that the (corresponding) sequence of cost approximating mappings is given by a continuous gradient mapping on $X \times X$. Tseng [67] establishes a convergence result for the nonlinear proximal descent algorithm similar to the result for Rule A; he, however, also assumes that the cost approximating mappings are given by a continuous mapping on $X \times X$.

One instance of the cyclic CA algorithm using Rule R is the cyclic gradient projection algorithm of Bertsekas and Tsitsiklis [11, Sec. 3.3.4]; it is identified by the choices $\varphi_i(x_i) = 1/(2\gamma)\|x_i\|^2$ and $\ell_t^i \equiv 1$, $i \in \mathcal{C}$, where γ is a sufficiently small positive constant [actually such that (22) is satisfied]. (See also Table 1.)

2.5. Linear convergence

The last result of this section establishes that, in the cyclic CA algorithm using Rule R with $\ell_t^i \equiv 1$, $\{x^t\}$ converges $|\mathcal{C}|$-step R-linearly to an element x^* of Ω, that is,

$$\|x^{t+|\mathcal{C}|} - x^*\| \leq c\rho^t, \qquad c > 0, \quad \rho \in (0,1), \qquad t = 0, 1, \ldots,$$

while $\{f(x^t)\}$ converges $|\mathcal{C}|$-step Q-linearly to $f(x^*)$, that is,

$$\limsup_{t \to \infty} \frac{f(x^{t+|\mathcal{C}|}) - f(x^*)}{f(x^t) - f(x^*)} = q < 1.$$

The result is a consequence of a relationship between a subclass of the CA algorithms and the algorithm framework of Luo and Tseng [43].

Assume that ∇f is Lipschitz continuous on X; in the sequential CA algorithm, let the sequence $\{i_t\}$ of indices be chosen according to the cyclic rule (7), $\{\Phi^t\}$ be chosen as in Theorem 6, and the step lengths be chosen according to Rule R with $\ell_t^i \equiv 1$. Also, we shall let one iteration correspond to a full cycle of $|\mathcal{C}|$ iterations in the algorithm described in Table 2.

It is straightforward to show that with these assumptions present,

$$x^{t+1} = P_X(x^t - \nabla f(x^t) + e^t), \qquad t = 0, 1, \ldots, \tag{23}$$

where P_X denotes Euclidean projection onto X and

$$e_i^t = x_i^{t+1} - x_i^t + \Phi_i^t(x_i^t) - \Phi_i^t(x_i^{t+1}) + \nabla_i f(x^t) - \nabla_i f(x_{i_-}^{t+1}, x_i^t, x_{i_+}^t), \qquad i \in \mathcal{C}.$$

It follows from the Lipschitz continuity of Φ_i^t and ∇f that

$$\|e_i^t\| \le (1 + M_{\Phi_i^t})\|x_i^t - x_i^{t+1}\| + M_{\nabla f}\|x^{t+1} - x^t\|, \qquad i \in \mathcal{C},$$

and hence, with $M_{\Phi_i} := \limsup_{t\to\infty}\{M_{\Phi_i^t}\}$,

$$\|e^t\| \le \left[1 + \sqrt{|\mathcal{C}|}\left(\max_{i\in\mathcal{C}}\{M_{\Phi_i}\} + M_{\nabla f}\right)\right]\|x^t - x^{t+1}\|, \qquad t = 0, 1, \ldots. \tag{24}$$

Assuming that $m_{\Phi_{i_t}^t} > M_{\nabla_{i_t} f}/2$ holds for all t, Rule R may be used with $\ell_t^i \equiv 1$ [cf. (22)]. We further assume that

$$m_{\Phi_i} := \liminf_{t\to\infty}\{m_{\Phi_i^t}\} > M_{\nabla_i f}/2, \qquad i \in \mathcal{C}. \tag{25}$$

Letting $\gamma := \min_{i\in\mathcal{C}}\{m_{\Phi_i} - M_{\nabla_i f}/2\} > 0$, summing the inequalities (6) [with $\ell_t^i \equiv 1$] for all $i \in \mathcal{C}$, we obtain that

$$f(x^{t+1}) - f(x^t) \le -\gamma\|x^t - x^{t+1}\|^2, \qquad t = 0, 1, \ldots. \tag{26}$$

From (23), (24), and (26), we may thus conclude that the sequential CA algorithm under the cyclic rule and using unit steps is of the form given by Luo and Tseng [43]. In order to apply their linear convergence result, we must further introduce two technical assumptions.

Assumption 2 (Local error bound) *For every $\nu \ge \inf_{x\in X} f(x)$ there are constants $\delta > 0$ and $\tau > 0$ such that*

$$\min_{y\in\Omega}\|x - y\| \le \tau\|x - P_X(x - \nabla f(x))\|$$

for all $x \in X$ with $f(x) \le \nu$ and $\|x - P_X(x - \nabla f(x))\| \le \delta$.

This assumption holds when f is strongly convex, or when X is polyhedral and f is either quadratic or a certain convex function. (See [43] for details.)

Assumption 3 (Proper separation of isocost surfaces) *There is a constant $\varepsilon > 0$ such that*

$$\|x - y\| \ge \varepsilon, \qquad \forall x, y \in \Omega, \quad f(x) \ne f(y).$$

This assumption holds whenever f takes a finite number of values on Ω; it holds in particular when f is convex.

Theorem 7 (Linear convergence of the cyclic CA algorithm under Rule R) *Assume that f is bounded from below on X and ∇f Lipschitz continuous on X. Let further Assumptions 2 and 3 hold. For each $i \in \mathcal{C}$ and t, let Φ_i^t be strongly monotone on X_i and Lipschitz continuous on X_i. Let further $m_{\Phi_i^t} > M_{\nabla_i f}/2$ hold for all $i \in \mathcal{C}$ and t, (25) hold and $M_{\Phi_i} < +\infty$ hold for all $i \in \mathcal{C}$. In the sequential CA algorithm, let the sequence $\{i_t\}$ be chosen according to the cyclic* *rule (7), and Rule R be used with $\ell_t^i \equiv 1$. Then, $\{f(x^t)\}$ converges at least $|\mathcal{C}|$-step Q-linearly and $\{x^t\}$ converges at least $|\mathcal{C}|$-step R-linearly to an element of Ω.*

Proof: Under the given assumptions, it follows from the above that the cyclic CA algorithm is of the form (23), (24), (26). The result then follows from Theorem 3.1 of [43]. ■

Luo and Tseng [43] establish linear convergence of the Gauss–Seidel scheme using their general result. Theorem 7 shows that a much larger class of cyclic decomposition algorithms enjoys a linear convergence rate. We conjecture that the essentially cyclic CA algorithm can be shown to be N-step linearly convergent under the assumptions of the theorem.

3. Synchronized parallel cost approximation algorithms

This section deals with a parallel decomposition version of the CA algorithm, in which it is assumed that the updating of the variable components is synchronized. Under this assumption, the resulting sequence of iterates will coincide with that of the original CA algorithm for particular choices of the cost approximating mappings Φ^t; the convergence characteristics of the algorithm is thus, in many cases, already known from the previous analyses made in [54, 55].

3.1. Synchronized parallel computations

We concentrate on the parallelism inherent in the subproblems of the CA algorithm, although there may be parallelism involved at lower levels in the computations (such as in the evaluations of function and gradient values). We envisage the existence of a multiprocessor powerful enough to be able to solve the $|\mathcal{C}|$ subproblems $[\mathrm{VIP}_{\Phi_i^t}]$ in parallel. (If fewer than $|\mathcal{C}|$ processors are available, then either some of the subproblems are solved in sequence or, if possible, the number of components is decreased; in either case, the analysis is the same, with the only exception that the value of $|\mathcal{C}|$ may change.)

If the subproblems $[\mathrm{VIP}_{\Phi_i^t}]$ are complex, as in the case of [TAP], then the parallel processors need to be powerful; since, in principle, each subproblem may also be solved using different algorithms, we are naturally led to consider a multiple-program-multiple-data (MPMD) algorithm [28]. (This is sometimes referred to as a *coarse-grained* parallel computer model.) The programming paradigm PVM (parallel virtual machine [27]) is based on such a computer model, as are hypercube computer architectures (e.g., the nCUBE [30]), and the CM-5.

The tasks, corresponding to the different subproblems, are allocated to the processors either before execution or at runtime. A static allocation is advantageous when the complexity of the tasks is uniform or predictable. When the tasks have unpredictable complexity, a dynamic allocation strategy improves the load balance among the processors [18]; this is done, however, at the expense of the overhead caused by the handling of the task queue [73].

We want to enforce the parallel CA algorithm to generate the same sequence as that of the original CA algorithm; we must then ensure that all the subproblems have been solved, the step length ℓ_t calculated, and the new iterate x^{t+1} communicated to all the processors, before iteration $t+1$ can start. This requires a *synchronization barrier* [34, 33, 11, 29] after the subproblem phase. The synchronized part of the algorithm (or, the critical section) may be implemented as follows. In a *shared memory* system [34, Sec. 7.1.2], each processor writes its result (y_i^t) on the main memory, after which it halts (unless there remains a task—a subproblem—in the queue to be allocated to that processor); when all subproblems have been solved, the step length ℓ_t and the new iterate x^{t+1} are calculated; the processors read this information from the memory and resume calculations on new subproblems. In a *distributed memory* system [11], the processors may, for example, send the solutions y_i^t through the message-passing system to a designated processor which is responsible for the calculation of ℓ_t; the resulting vector x^{t+1} is then broadcast to the processors which then commence iteration $t+1$. (The master processor could also be responsible for the construction of the new subproblems.)

The synchronization introduces several types of overhead: first, the execution of the synchronization barrier itself; second, the load imbalance among the processors means that some of the processors may be idle most of the time; third, the updating step is, in general, a serial operation whose contribution to the total computational task may be substantial if an accurate line search is made. The efficiency can be further degraded by memory conflicts (shared memory systems [35]) or slow communication channels (distributed systems [11]). These potential drawbacks motivate the study of asynchronous parallel algorithms in Section 4.

3.2. The synchronized parallel algorithm

We let the partition of $\Re^n$ be defined by (1). In the original CA algorithm, given an iteration t and a point $x^t \in X$, let the cost approximating mapping Φ^t be of the form (2). It then follows that the subproblem [VIP$_{\Phi^t}$] separates into $|\mathcal{C}|$ independent subproblems [VIP$_{\Phi_i^t}$], which may be solved in parallel under the assumption of this section. The *synchronized parallel cost approximation algorithm* is therefore nothing but the original CA algorithm, applied to the structured problem [CDP] and choosing cost approximating mappings that match the separability structure.

A summary of the synchronized parallel CA algorithm is given in Table 3.

Table 3. The synchronized parallel cost approximation algorithm

0. (*Initialization*): Choose an initial point $x^0 \in X$, and let $t = 0$.

1. (*Search direction generation*): Find a solution y_i^t to [VIP$_{\Phi_i^t}$] for each $i \in \mathcal{C}$. The resulting search direction is $d^t = y^t - x^t$.

2. (*Termination criterion*): If x_i^t solves [VIP$_{\Phi_i^t}$] for all $i \in \mathcal{C} \to$ Stop ($x^t \in \Omega$). Otherwise, continue.

3. (*Line search*): Choose a step length, ℓ_t, such that $x^t + \ell_t d^t \in X$ and the value of f is reduced sufficiently.

4. (*Update*): Let $x^{t+1} = x^t + \ell_t d^t$, and $t := t+1$.

5. (*Termination criterion*): If x^t is acceptable as a solution $\to$ Stop. Otherwise, go to Step 1.

The algorithms presented in Table 1 all have natural parallel decomposition versions.

In the parallel algorithm, choose the cost approximating mapping of the form (2), where

$$\Phi_i^t = \nabla_i f(x_{i_-}^t, \cdot, x_{i_+}^t), \qquad i \in \mathcal{C}. \tag{27}$$

This mapping is monotone on X whenever f is convex in each component x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, $i \in \mathcal{C}$, it follows that the subproblem [VIP$_{\Phi^t}$] is equivalent (under the above convexity assumption on f) to finding a

$$y_i^t \in \arg\min \{ f(x_{i_-}^t, y_i, x_{i_+}^t) \mid y_i \in X_i \}, \qquad i \in \mathcal{C}.$$

Choosing $\ell_t = 1$ yields $x^{t+1} = y^t$, and the resulting iteration is that of the *Jacobi* algorithm [50, 11] (also known as the *method of simultaneous displacements*). The Jacobi algorithm, which was originally proposed for the solution of systems of equations, is therefore a parallel CA algorithm where the cost approximating mapping is (27) and unit steps are used.

Each component of the mapping Φ^t is of the form (8), which shows the strong relationships to the Gauss–Seidel algorithm; the difference is, of course, that all variable components here are updated simultaneously. The lack of a line search in the Jacobi algorithm (the term *successive approximation* is often used to describe an algorithm in which $\ell_t \equiv 1$ is set) implies the need for very strong assumptions on f in order to establish convergence (see, e.g., [11, Prop. 3.3.10] and [28, Chap. 8]). The first presentation of a Jacobi algorithm including line searches is [24]. (The algorithms of [15, 36] are also highly related to the Jacobi method; see [53] and [56, Sec. 4.3.3] for further discussions on Jacobi type methods applied to the traffic assignment problem.) It is clear that the introduction of a line search in Jacobi type algorithms may enhance the practical performance of the algorithm as well as ensuring its convergence under weaker conditions (see, e.g., [49]).

3.3. Convergence of the conceptual algorithm

We let the sequence $\{\Phi^t\}$ of cost approximating mappings be defined by the gradient mapping of the function $\varphi : X \times X$ given by (9); the convergence conditions are identical to those of Theorems 1 and 3.

THEOREM 8 (Convergence of the synchronized parallel CA algorithm under Rule E) *Let the sequence $\{\Phi^t\}$ of cost approximating mappings be given by the gradient of a continuous function $\varphi : X \times X \mapsto \Re$ of the form (9), where $\varphi_i(\cdot, x)$ is convex and in C^1 on X_i. In the synchronized parallel CA algorithm, let Rule E be used. Assume that $x^0 \in X$ is such that the lower level set $L(x^0)$ is bounded, and that the problem $[\mathrm{CDP}_{\varphi_i}]$ is well defined, in the sense that the set $Y_i(x)$ is nonempty and bounded for all $x \in L(x^0)$. Then, $\{f(x^t)\} \to f(\overline{x})$ for some $\overline{x} \in \Omega$, any accumulation point of the sequence $\{x^t\}$ (at least one such point exists) lies in Ω, and (10) holds.*

Proof: The algorithm is identical to the original CA algorithm, with the choice (9) of function φ. The result then follows immediately from Theorem 4.2 of [54]. ∎

3.4. Truncated synchronized parallel cost approximation

The truncated synchronized parallel CA algorithm is obtained from replacing Step 1 of the synchronized parallel CA algorithm (see Table 3) by (cf. Section 2.3):

1". (*Search direction generation*): For each $i \in \mathcal{C}$, apply $1 \leq k_i^t \leq \overline{k}_i$ iterations of the algorithm described by B_i, starting from x_i^t, that is, let

$$(\overline{y}_i^t, x^t) \in \underbrace{B_i \circ \ldots \circ B_i}_{k_i^t \text{ times } B_i}(x_i^t, x^t).$$

The resulting search direction is $d^t = \overline{y}^t - x^t$.

Since the values of k_i^t are allowed to vary among the components $i \in \mathcal{C}$, they may be used in a scheme for enhancing the load balance among the processors; one of the many possibilities is to terminate the solution of *all* the subproblems $[\mathrm{VIP}_{\Phi_i^t}]$ whenever a sufficiently large portion of them have already been terminated through the use of other termination criteria.

The following result has the same conditions as those of Theorem 2 and 4.

THEOREM 9 (Convergence of the truncated synchronized parallel CA algorithm) *Replace Step 1 by Step 1" in the synchronized parallel CA algorithm. In Theorem 8, replace the condition that the set $Y_i(x)$ is nonempty and bounded for all $x \in L(x^0)$ with Assumption 1. Then, the conclusions of Theorem 8 hold for the truncated synchronized parallel CA algorithm.*

Proof: The result is established in a manner similar to that of the proof of Theorem 2. We extract positive integers k_i, from which we construct the mappings

$$\overline{Y}_i(x) = \{\, \overline{y}_i \in \Re^{n_i} \mid (\overline{y}_i, x) \in \underbrace{B_i \circ \ldots \circ B_i}_{k_i \text{ times } B_i}(x_i, x) \,\}, \qquad i \in \mathcal{C};$$

the algorithmic map is $A = E \circ \prod_{i \in \mathcal{C}} \overline{D}_i$, where $\overline{D}_i(x) = \overline{Y}_i(x) - x_i$, $i \in \mathcal{C}$. This mapping can be shown to satisfy the conditions of Zangwill's Theorem. The mapping C in the Spacer Step Theorem is identified as in the proof of Theorem 2, as is the remainder of the proof. ∎

3.5. Synchronized parallel cost approximation under different step length rules

We next replace the exact line search by the Armijo step length rule.

THEOREM 10 (Convergence of the synchronized parallel CA algorithm under Rule A) *Let the sequence $\{\Phi^t\}$ of cost approximating mappings be either given by a continuous mapping on $X \times X$ of the form (15), where $\Phi_i(\cdot, x)$ is maximal monotone on X_i and strictly monotone on X_i, or by the gradient of a continuous function $\varphi : X \times X \mapsto \Re$ of the form (9), where $\varphi_i(\cdot, x)$ is convex and in C^1 on X_i. In the synchronized parallel CA algorithm, let Rule A be used. Assume that $x^0 \in X$ is such that the lower level set $L(x^0)$ is bounded. Assume that the problem* [VIP$_{\Phi_i}$] *(respectively,* [CDP$_{\varphi_i}$]*) is well defined, in the sense that (16) holds [respectively, the set $Y_i(x)$ is nonempty and bounded for all $x \in L(x^0)$]. Then, $\{f(x^t)\} \to f(\overline{x})$ for some $\overline{x} \in \Omega$, any accumulation point of the sequence $\{x^t\}$ (at least one such point exists) lies in Ω, and (10) holds.*

Proof: Similarly to the proof of Theorem 5, we obtain that $\{\nabla f(x^t)^{\mathrm{T}} d^t\} \to 0$ holds. Let x^∞ and y^∞ be limit points of the sequences $\{x^t\}$ and $\{y^t\}$, respectively. If Φ is strictly monotone, then the limit of [VIP$^t_\Phi$] implies that $x^\infty \in \Omega$. If $\Phi \equiv \nabla_y \varphi$, then, by Lemma 1.d, in the limit of [VIP$^t_{\nabla\varphi}$], $y^\infty \in Y(x^\infty)$ holds. But by the relation $\nabla f(x^\infty)^{\mathrm{T}}(y^\infty - x^\infty) = 0$ and the convexity of φ, $f_\varphi(y^\infty) = f(x^\infty)$ actually holds. Since $f_\varphi(x^\infty) = f(x^\infty)$ holds by construction, $x^\infty \in Y(x^\infty)$, and hence, by Lemma 1.a.i, $x^\infty \in \Omega$. The rest of the proof follows that of Theorem 1. ∎

Compared to Theorem 5, it is here possible to establish convergence under mere monotonicity of the cost approximating mapping, provided that it is a gradient.

We next combine the truncated algorithm with the Armijo step length rule, and thus provide an implementable algorithm.

THEOREM 11 (Convergence of the truncated synchronized parallel CA algorithm under Rule A) *In the truncated synchronized parallel CA algorithm of Theorem 9, let Rule A be used. Assume that the sequence $\{\overline{y}^t\}$ of truncated subproblem solutions is bounded. Then, the conclusions of Theorem 9 hold for the truncated synchronized parallel CA algorithm under the Armijo step length rule.*

Proof: The Armijo rule is valid (cf. Lemma 1.e.ii, Lemma 2.a.ii and Assumption 1.4). Continuing as in the proof of the previous theorem (using the boundedness assumption on $\{\overline{y}^t\}$), we may conclude that $\{\nabla f(x^t)^{\mathrm{T}} d^t\} \to 0$ holds. Let x^∞ and $\overline{y}^\infty$ be limit points of the sequences $\{x^t\}$ and $\{\overline{y}^t\}$, respectively. Then, $f_\varphi(\overline{y}^\infty) \le f_\varphi(x^\infty)$ holds. Combining this relation with the convexity of f_φ and $\nabla f(x^\infty)^{\mathrm{T}}(y^\infty - x^\infty) = 0$, we obtain that

$$f_\varphi(\overline{y}^\infty) = f_\varphi(x^\infty). \tag{28}$$

We next extract, for each $i \in \mathcal{C}$, a number, k_i, that occurs an infinite number of times in the bounded sequence $\{k_i^t\}$, and consider the corresponding subsequence of $[\mathrm{CDP}_{\varphi_i}^t]$. In the limit of this subsequence, $[\mathrm{CDP}_{\varphi_i}^t]$ is solved using k_i steps of the algorithm B_i. By arguments similar to those in the proof of Theorem 2, Assumption 1 yields that

$$(\overline{y}_i^\infty, x^\infty) \in \underbrace{B_i \circ \ldots \circ B_i}_{k_i \text{ times } B_i}(x_i^\infty, x^\infty).$$

Using (28), Lemma 1.e.ii and Assumption 1.4, we may conclude that $x^\infty \in Y(x^\infty)$, that is, that $x^\infty \in \Omega$. The rest of the proof follows that of Theorem 1. ■

Note that the sequence $\{\overline{y}^t\}$ produced by the truncated subproblem algorithm need not be bounded under Assumption 1 in general (under the exact step length Rule E, this is however not a necessary property); a sufficient condition is, however, that X is bounded.

THEOREM 12 (Convergence of the synchronized parallel CA algorithm under strongly monotone mappings Φ^t) *Assume that f is bounded from below on X. For each t, let the cost approximating mapping be of the form (2), where each component mapping Φ_i^t is strongly monotone on X_i and Lipschitz continuous on X_i. Let further $m_{\Phi_i} := \liminf_{t\to\infty}\{m_{\Phi_i^t}\} > 0$ and $M_{\Phi_i} := \limsup_{t\to\infty}\{M_{\Phi_i^t}\} < +\infty$ hold for all $i \in \mathcal{C}$. Then, in the synchronized parallel CA algorithm, from any starting point $x^0 \in X$, under Rules E and A, any accumulation point of $\{x^t\}$ lies in Ω.*

Assume further that ∇f is Lipschitz continuous on X. If, in Rule R,

$$\ell_t \in \left[\varepsilon, \min\left\{\overline{\ell}_t, 2\min_{i\in\mathcal{C}}\{m_{\Phi_i^t}\}/M_{\nabla f} - \varepsilon\right\}\right], \qquad \varepsilon \in \left(0, \min_{i\in\mathcal{C}}\{m_{\Phi_i}\}/M_{\nabla f}\right],$$

then the same conclusion holds for Rule R.

If further $\{x^t\}$ is bounded and Ω is nonempty, then for the Rules E, A and R, (10) holds.

If further Ω is finite, then for the Rules E, A and R, $\{x^t\}$ converges to one point in Ω.

Proof: Similarly to the proof of Theorem 6, we obtain that $\{d^t\} \to 0$. From $[\mathrm{VIP}_{\Phi_i^t}]$, we obtain that

$$\begin{aligned} \nabla_i f(x^t)^{\mathrm{T}}(y_i - y_i^t) &\geq -[\Phi_i^t(y_i^t) - \Phi_i^t(x_i^t)]^{\mathrm{T}}(y_i - y_i^t) \\ &\geq -M_{\Phi_i^t}\|y_i^t - x_i^t\| \cdot \|y_i - y_i^t\|, \qquad \forall y_i \in X_i, \qquad i \in \mathcal{C}. \end{aligned}$$

Let x^∞ be an accumulation point of the sequence $\{x^t\}$. The above then implies that, in the limit of the corresponding subsequence,

$$\nabla_i f(x^\infty)^{\mathrm{T}}(y_i - x_i^\infty) \geq 0, \qquad \forall y_i \in X_i, \qquad i \in \mathcal{C},$$

that is, $x^\infty \in \Omega$. The rest of the proof follows that of Theorem 6. ■

Remark. We note that the Lipschitz continuity assumption for Rule R here is stronger than that in Theorem 6 for the corresponding sequential algorithm, the reason being that the update here is performed over all components of x simultaneously.

In the sequential algorithm, the individual step lengths in Rule R must satisfy

$$\ell_t^i < \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}, \qquad i \in \mathcal{C}, \qquad t = 0, 1, \ldots;$$

from the relation $M_{\nabla_i f} \leq M_{\nabla f}$ follows that

$$\frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j^t}\}}{M_{\nabla f}} \le \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}, \qquad i\in\mathcal{C}, \qquad t=0,1,\ldots, \tag{29}$$

and we conclude that, in general, longer steps are allowed in the sequential algorithm than in the synchronized parallel algorithm. (The same observation is made by Cohen [16, 17] for the auxiliary problem principle.) One may therefore expect the sequential algorithm to converge to a solution with a given accuracy in less iterations, although the parallel algorithm may be more efficient in terms of solution time. The intuitive explanation for this observation is that the sequential algorithm utilizes more recent information in the construction of the subproblems.

3.6. Linear convergence

THEOREM 13 (Linear convergence of the synchronized parallel CA algorithm under Rule R) *Assume that f is bounded from below on X and ∇f Lipschitz continuous on X. Let further Assumptions 2 and 3 hold. For each $i\in\mathcal{C}$ and t, let Φ_i^t be strongly monotone on X_i and Lipschitz continuous on X_i. Let further $\min_{i\in\mathcal{C}}\{m_{\Phi_i^t}\} > M_{\nabla f}/2$ hold for all t, $m_{\Phi_i} := \liminf_{t\to\infty}\{m_{\Phi_i^t}\}$, $i\in\mathcal{C}$, satisfy $\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} > M_{\nabla f}/2$, and $M_{\Phi_i} := \limsup_{t\to\infty}\{M_{\Phi_i^t}\} < +\infty$ for all $i\in\mathcal{C}$. In the synchronized parallel CA algorithm, let Rule R be used with $\ell_t\equiv 1$. Then $\{f(x^t)\}$ converges at least Q-linearly and $\{x^t\}$ converges at least R-linearly to an element of Ω.*

Proof: The algorithm is of the form (23), where $e^t = x^{t+1} - x^t + \Phi^t(x^t) - \Phi^t(x^{t+1})$. The relation (24) here corresponds to $\|e^t\| \le (1+\max_{i\in\mathcal{C}}\{M_{\Phi_i}\})\|x^t - x^{t+1}\|$, and (26) holds with $\gamma = \min_{i\in\mathcal{C}}\{m_{\Phi_i}\} - M_{\nabla f}/2 > 0$. The result then follows from Theorem 3.1 of [43]. ■

Although the parallel version of the algorithm may speed-up the practical convergence rate compared to the sequential one, the need for synchronization for carrying out the updating step may still deteriorate the performance, since faster processors must wait for slower ones. In the next section, we therefore consider an asynchronous version of the parallel algorithm, in which processors do not wait to receive the latest information available.

4. Partially asynchronous parallel cost approximation algorithms

In the algorithms considered in this section, the synchronization among the processors is removed. Because the speed of computations and communications can vary among the processors, and communication delays can be substantial, the processors will perform the calculations out of phase with each other. Thus, the advantage of a reduced synchronization penalty is paid for by an increase in interprocessor communications, a use of outdated information that may be counterproductive if certain conditions are not met, and a more difficult convergence detection (see [11]). (Certainly, the convergence analysis also becomes more complicated.) Recent numerical experiments indicate, however, that the introduction of such *asynchronous* computations can substantially enhance the efficiency of parallel iterative methods (e.g., [23, 8, 13]).

4.1. Asynchronous parallel computations

The convergence of asynchronous iterative methods has been studied by several researchers (e.g., [14, 45, 3, 6, 11, 72, 46, 73]). We distinguish between two asynchronous computing models. A *totally asynchronous*, or *chaotic*, algorithm model allows arbitrarily large communication delays and differences in the frequency of computation of different processors; examples of algorithms that have been analyzed using this model are dynamic programming methods, fixed point methods, and coordinate search methods for linear and convex network flows (e.g., [5, 6, 10, 9, 11, 7]). A *partially asynchronous* algorithm model assumes the existence of an upper bound on the communication delays; this has been the main computing model by which asynchronous gradient-type algorithms have been analyzed (see [11, Sec. 6.3.2]). Partially asynchronous versions of instances of the CA algorithm have, however, been studied only to a limited extent; variable metric (or, deflected gradient) methods in unconstrained optimization [11, Sec. 7.5], gradient projection methods [71, 11, 68], and coordinate ascent methods for dual formulations of strictly convex network flow problems [70] are the only examples to date.

The model of partial asynchronism utilized is adapted from [11, Sec. 7.1]. For each processor (or, variable component) $i \in \mathcal{C}$, we introduce

(a) initial conditions, $x_i(t) = x_i^0 \in X_i$, for all $t \leq 0$,

(b) a set $\mathcal{T}^i$ of times at which x_i is updated, and

(c) a variable $\tau_j^i(t)$ for each $j \in \mathcal{C}$ and $t \in \mathcal{T}^i$, denoting the time at which the value of x_j used by processor i at time t is generated by processor j, satisfying $0 \leq \tau_j^i(t) \leq t$ for all $j \in \mathcal{C}$ and $t \geq 0$.

We note the interesting fact that the two computational models presented previously actually belong to the class of asynchronous algorithms: the cyclic sequential algorithm model is obtained from the choices $\mathcal{T}^i = \cup_{k \geq 0}\{|\mathcal{C}|k + i - 1\}$ and $\tau_j^i(t) = t$, while the synchronous parallel model is obtained by choosing $\mathcal{T}^i = \mathcal{Z}_+$ and $\tau_j^i(t) = t$, for all i, j and t.

The communication delay from processor j to processor i at time t is $t - \tau_j^i(t)$. The convergence of the *partially* asynchronous CA algorithm is based on the assumption that this delay is upper bounded.

ASSUMPTION 4 (Partial asynchronism) *There exists a positive integer P such that*

(i) *for every $i \in \mathcal{C}$ and $t \geq 0$, at least one element of $\{t, \dots, t + P - 1\}$ belongs to $\mathcal{T}^i$, and*

(ii) $0 \leq t - \tau_j^i(t) \leq P - 1$ *holds for all $i, j \in \mathcal{C}$ and all $t \geq 0$.*

In short, the assumption states that no processor waits for an arbitrarily long time to compute a subproblem solution or to receive a message from another processor. (Note that a synchronized model satisfies $P = 1$.) For further discussions on the assumption, we refer to [11, 70]; we only remark that it is often easily enforced in a practical implementation.

The iterate $x(t)$ is defined by the vector of $x_i(t)$, where $x_i(t)$ is updated by processor i according to

$$x_i(t+1) = x_i(t) + \ell(y_i(t) - x_i(t)), \qquad i \in \mathcal{C}, \qquad t \in \mathcal{T}^i, \tag{30}$$

that is, by using Rule R with a fixed step length ℓ. We also let $d_i(t) := y_i(t) - x_i(t) = 0$ for all $t \notin \mathcal{T}^i$. Processor i has knowledge of a possibly outdated version of $x(t)$; we let

$$x^i(t)^{\mathrm{T}} = \left[x_1(\tau_1^i(t))^{\mathrm{T}}, \dots, x_{|\mathcal{C}|}(\tau_{|\mathcal{C}|}^i(t))^{\mathrm{T}}\right]$$

denote this vector.

4.2. Partially asynchronous parallel cost approximation

The next result is an extension of Theorem 12 for Rule R. The use of outdated information introduces error terms which, by the existence of the bound P, however can be made insignificant by choosing the step length ℓ small enough. (We must also build the sequence of cost approximating mappings based on a continuous mapping on $X \times X$.)

THEOREM 14 (Convergence of the partially asynchronous parallel CA algorithm under Rule R) *Assume that f is bounded from below on X and ∇f Lipschitz continuous on X. Let the sequence $\{\Phi^t\}$ of cost approximating mappings be given by a continuous mapping on $X \times X$ of the form (15), where $\Phi_i(\cdot, x)$ is strongly monotone on X_i and Lipschitz continuous on X_i. Let Assumption 4 hold, with $P > 1$. In the partially asynchronous parallel CA algorithm, let Rule R be used with a fixed step length ℓ. Then, if $\ell \leq 1$ and*

$$\ell \in \left(0, \frac{2\min_{i\in\mathcal{C}}\{m_{\Phi_i}\}}{M_{\nabla f}[1+(|\mathcal{C}|+1)P]}\right), \tag{31}$$

from any starting point $x(0) \in X$, any accumulation point of the sequence $\{x(t)\}$ lies in Ω. If further $\{x(t)\}$ is bounded and Ω is nonempty, then (10) holds. If further Ω is finite, then $\{x(t)\}$ converges to one point in Ω.

Proof: By the Lipschitz continuity of ∇f and the strong monotonicity of $\Phi_i(\cdot, x)$, $i \in \mathcal{C}$,

$$\begin{aligned}
f(x(t+1)) - f(x(t)) &\leq \ell \sum_{i\in\mathcal{C}} \nabla_i f(x(t))^{\mathrm{T}} d_i(t) + (M_{\nabla f}/2\ell^2)\|d(t)\|^2 \\
&= \ell \sum_{i\in\mathcal{C}} \nabla_i f(x^i(t))^{\mathrm{T}} d_i(t) + (M_{\nabla f}/2\ell^2)\|d(t)\|^2 \\
&\quad + \ell \sum_{i\in\mathcal{C}} \left[\nabla_i f(x(t)) - \nabla_i f(x^i(t))\right]^{\mathrm{T}} d_i(t) \qquad (32) \\
&\leq \ell \left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2\ell\right) \|d(t)\|^2 \\
&\quad + M_{\nabla f}\ell \sum_{i\in\mathcal{C}} \|d_i(t)\| \cdot \|x(t) - x^i(t)\|. \qquad (33)
\end{aligned}$$

Next, we bound $\|x(t) - x^i(t)\|$, $i \in \mathcal{C}$. From Assumption 4.ii and the use of Rule R,

$$\|x_j^i(t) - x_j(t)\| = \|x_j(\tau_j^i(t)) - x_j(t)\| = \ell \left\| \sum_{\tau=\tau_j^i(t)}^{t-1} d_j(\tau) \right\| \leq \ell \sum_{\tau=t-P}^{t-1} \|d_j(\tau)\|, \qquad i, j \in \mathcal{C}.$$

By the triangle inequality,

$$\|x^i(t) - x(t)\| \leq \ell \sum_{\tau=t-P}^{t-1} \|d(\tau)\|, \qquad i \in \mathcal{C}. \tag{34}$$

Using the inequality (34) in (33),

$$\begin{aligned}
f(x(t+1)) - f(x(t)) &\leq \ell \left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2\ell\right) \|d(t)\|^2 \\
&\quad + M_{\nabla f}\ell^2 \sum_{i\in\mathcal{C}} \|d_i(t)\| \sum_{\tau=t-P}^{t-1} \|d(\tau)\|.
\end{aligned}$$

We then use the inequality $2\|d_i(t)\| \cdot \|d(\tau)\| \leq \|d_i(t)\|^2 + \|d(\tau)\|^2$, $i \in \mathcal{C}$, to obtain

$$\begin{aligned} f(x(t+1)) - f(x(t)) &\leq \ell \left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2\ell \right) \|d(t)\|^2 \\ &\quad + M_{\nabla f}/2\ell^2 \sum_{i\in\mathcal{C}} \sum_{\tau=t-P}^{t-1} \left(\|d_i(t)\|^2 + \|d(\tau)\|^2 \right) \\ &= \ell \left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2[1+P]\ell \right) \|d(t)\|^2 \\ &\quad + |\mathcal{C}| M_{\nabla f} \ell^2 \sum_{\tau=t-P}^{t-1} \|d(\tau)\|^2. \end{aligned}$$

Adding this inequality over all t yields

$$f(x(t+1)) - f(x(0)) \leq \ell \left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2[1+P(|\mathcal{C}|+1)]\ell \right) \sum_{\tau=0}^{t} \|d(\tau)\|^2,$$

where we let $d(t) = 0$ for all $t < 0$.

If (31) holds, then $\{d(t)\} \to 0$ must hold. We also have from (30) that

$$\{x(t+1) - x(t)\} \to 0, \tag{35}$$

and from (34) that

$$\{x^i(t) - x(t)\} \to 0. \tag{36}$$

Let x^∞ be any accumulation point of the sequence $\{x(t)\}$, corresponding to a convergent subsequence $\{t_k\}$, and let τ_k be such that $|t_k - \tau_k| \leq P$ and $\tau_k \in \mathcal{T}^i$ (cf. Assumption 4.i). Then, (35) and (36) imply that $\{x(\tau_k)\} \to x^\infty$ and $\{x^i(\tau_k)\} \to x^\infty$ holds. From (30) it finally holds that $y^\infty = x^\infty$, that is, $x^\infty \in \Omega$. The last statement follows as in Theorem 6. ∎

We conjecture that this algorithm is actually linearly convergent under the assumptions of the theorem (although possibly for a smaller value of the maximal step length); the reason for this belief is that Tseng [68] has recently established the linear convergence of the special case of partially asynchronous parallel gradient projection.

4.3. Qualitative analysis

The upper bound

$$\bar{\ell} = \frac{2\min_{i\in\mathcal{C}}\{m_{\Phi_i}\}}{M_{\nabla f}[1 + (|\mathcal{C}|+1)P]} \tag{37}$$

on the maximal step length allowed provides interesting information on the relationships between the properties of the problem, as quantified by the constant $M_{\nabla f}$, the maximal allowed asynchronism, P, the number of independent processors used (or, variable components), $|\mathcal{C}|$, and the resulting maximal step length.

Most important to note is that $\bar{\ell}$ is (essentially) inversely proportional to the maximal allowed asynchronism P; this is very intuitive, since if processors take longer steps then they should exchange information more often. Conversely, the more outdated the information is, the less reliable it is, hence the shorter step. Note that when $P = 1$, that is, when the algorithm is synchronized, the right–hand–side of (34) is zero; by tracing the proof, the resulting upper bound can be shown to equal that in the synchronized approach (Theorem 12).

Keeping $\overline{\ell}$ fixed, (37) states that P should be smaller if $M_{\nabla f}$ increases; this is natural since ∇f then changes more rapidly, and a smaller change in $x_i(t)$ is tolerated before the other processors receives new information from processor i. (This is true also for the other two algorithm models.)

A similar relationship holds between $\overline{\ell}$ and the number $|\mathcal{C}|$ of processors utilized. This is also intuitive, since the more processors that are being utilized, the more information needs to be distributed, and hence the communication delay increases. Although this result indicates that the amount of parallelism should be kept down to a minimum, the best choice of definition of $|\mathcal{C}|$ depends both on the complexity of the subproblems [VIP$_{\Phi_i}$] and on the speed-up obtained from an increased amount of parallelism.

Summarizing the results obtained for the three algorithm versions, (29) and (31) imply, for $P > 1$, that

$$\frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j}\}}{M_{\nabla f}[1+(|\mathcal{C}|+1)P]} < \frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j^t}\}}{M_{\nabla f}} \leq \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}, \qquad i\in\mathcal{C}, \qquad t=0,1,\ldots. \tag{38}$$

The relations in (38) quantify the intuitive result that utilizing an increasing degree of parallelism and asynchronism results in a decreasing quality of the step directions, due to the usage of more outdated information; subsequently, smaller step lengths must be used.

Bertsekas and Tsitsiklis [11, Sec. 7.5.2] discuss the role of the above parameters for the special case of deflected gradient algorithms in unconstrained optimization. They also introduce a refinement of the Lipschitz continuity assumption, in order to describe the effect of the degree of coupling between the independent components x_i in the objective function on the maximal asynchronism allowed. Assume that there is a positive constant M with

$$\|\nabla_i f(x) - \nabla_i f(y)\| \leq M\|x-y\|, \qquad \forall x,y\in X \quad \text{with} \quad x_i = y_i, \qquad i\in\mathcal{C}.$$

If f is separable with respect to the partition of $\Re^n$ defined by (1), then $M = 0$ holds; a small value of M indicates that the problem is weakly coupled. (Note that $M \leq M_{\nabla f}$ holds.) We also include, in Assumption 4, the condition that

(iii) $\tau_i^i(t) = t, \qquad t\in\mathcal{T}^i, \qquad i\in\mathcal{C}.$

This assumption states that processor i always uses the most recent value of its own component x_i of x, and is in [73] referred to as a computational *non-redundancy* condition. This condition holds in general when no variable component is updated simultaneously by more than one processor.

These new assumptions imply that $x_i(t) = x_i^i(t)$ holds for all $i\in\mathcal{C}$, and the second occurrence of the constant $M_{\nabla f}$ in the right–hand–side of the inequality (33) may be replaced by the smaller constant M. The upper bound on the step length (assuming that $P > 1$) then is

$$\overline{\ell} = \frac{2\min_{i\in\mathcal{C}}\{m_{\Phi_i}\}}{M_{\nabla f}+(|\mathcal{C}|+1)MP}.$$

Thus, for a fixed step length, the asynchronism should be (essentially) inversely proportional to the degree of coupling between the variable components in the objective function.

5. Concluding remarks

The purpose of this paper is to give a unified presentation of iterative descent algorithms for problems defined over Cartesian product sets. Three decomposition versions of the cost approximation algorithm are presented and established. They are all well motivated, for different reasons: depending on the problem size and the computer facilities available, parallel implementations of the algorithm may be out of the question, and a sequential algorithm the only practical alternative; with the proper computer facilities, however, synchronized or partially asynchronous parallel algorithms may be used. The convergence analysis shows under which circumstances the three versions produce the correct output.

The convergence results for the sequential and partially asynchronous parallel models can be extended by allowing the variable components x_i, $i \in \mathcal{C}$, to overlap. In the sequential algorithm, for example, this possibility can be used to allow some variable components to be updated more often than others. In the partially asynchronous algorithm, this implies that, in the terminology of Üresin and Dubois [73], computations are *redundant*, that is, more than one processor may update the same variable component simultaneously.

Most of the results obtained in this paper may be extended to more general problems. The objective function may, for example, be non-differentiable, provided that the non-differentiability present is additively separable with respect to the partition of $\Re^n$ defined by (1), that is, the objective function is of the form

$$T(x) = f(x) + \sum_{i \in \mathcal{C}} u_i(x_i),$$

where $u_i : \Re^{n_i} \mapsto \Re \cup \{+\infty\}$ is a lower semicontinuous, proper and convex function with an effective domain $\mathtt{dom}\, u_i \supset X_i$, $i \in \mathcal{C}$ (see [54, Sec. 3.12]).

The algorithms in this paper may also be extended to asymmetric variational inequality problems over Cartesian product sets. In such a problem, we seek an $x^* \in X$ such that

$$F(x^*)^{\mathrm{T}}(x - x^*) \geq 0, \qquad \forall x \in X, \tag{39}$$

where $F : X \mapsto \Re^n$ is a continuous mapping which is neither the gradient of a real-valued function nor a mapping of the form $[\nabla_y L^{\mathrm{T}}, -\nabla_z L^{\mathrm{T}}]^{\mathrm{T}}$ for some saddle function $L : Y \times Z \mapsto \Re$ $(X = Y \times Z)$. The updating step of the CA algorithm in this paper is then realized through an (approximate) line search with respect to a merit function for (39); see [54, 37, 57] for more details.

Acknowledgements

The author wishes to thank an anonymous reviewer for valuable comments. This research was sponsored in part by a postdoctoral grant from the Swedish Research Council for Engineering Sciences (TFR) (no. 282-93-1195).

References

1. L. Armijo, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math., 16 (1966), pp. 1–3.
2. K. J. Arrow and L. Hurwicz (eds.), *Studies in Resource Allocation Processes*, Cambridge Uinversity Press, Cambridge, 1977.
3. G. M. Baudet, *Asynchronous iterative methods for multiprocessors*, J. ACM, 25 (1978), pp. 226–244.
4. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, New York, NY, second ed., 1993.
5. D. P. Bertsekas, *Distributed dynamic programming*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 610–616.
6. D. P. Bertsekas, *Distributed asynchronous computation of fixed points*, Math. Programming, 27 (1983), pp. 107–120.
7. D. P. Bertsekas, *Auction algorithms for network flow problems: a tutorial introduction*, Comput. Optim. Appl., 1 (1992), pp. 7–66.

8. D. P. Bertsekas and D. A. Castañon, *Parallel synchronous and asynchronous implementations of the auction algorithm*, Parallel Comput., 17 (1991), pp. 707–732.
9. D. P. Bertsekas and J. Eckstein, *Dual coordinate step methods for linear network flow problems*, Math. Programming, 42 (1988), pp. 203–243.
10. D. P. Bertsekas and D. El Baz, *Distributed asynchronous relaxation methods for convex network flow problems*, SIAM J. Control Optim., 25 (1987), pp. 74–85.
11. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, London, 1989.
12. Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAM Rev., 23 (1981), pp. 444–466.
13. E. D. Chajakis and S. A. Zenios, *Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization*, Parallel Comput., 17 (1991), pp. 873–894.
14. D. Chazan and W. Miranker, *Chaotic relaxation*, Linear Algebra Appl., 2 (1969), pp. 199–222.
15. R. J. Chen and R. R. Meyer, *Parallel optimization for traffic assignment*, Math. Programming, 42 (1988), pp. 327–345.
16. G. Cohen, *Optimization by decomposition and coordination: a unified approach*, IEEE Trans. Automat. Control, AC-23 (1978), pp. 222–232.
17. G. Cohen, *Auxiliary problem principle and decomposition of optimization problems*, J. Optim. Theory Appl., 32 (1980), pp. 277–305.
18. G. Cybenko, *Dynamic load balancing for distributed memory multiprocessors*, Technical Report 87-1, Department of Computer Science, Tufts University, Medford, MA, 1987.
19. G. B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Oper. Res., 8 (1960), pp. 101–111.
20. R. S. Dembo, J. M. Mulvey, and S. A. Zenios, *Large-scale nonlinear network models and their application*, Oper. Res., 37 (1989), pp. 353–372.
21. Y. M. I. Dirickx and L. P. Jennergren, *Systems Analysis by Multilevel Methods: With Applications to Economics and Management*, vol. 6 of International Series on Applied Systems Analysis, John Wiley & Sons, Chichester, 1979.
22. R. W. Eash, B. N. Janson, and D. E. Boyce, *Equilibrium trip assignment: advantages and implications for practice*, Transp. Res. Rec., 728 (1979), pp. 1–8.
23. D. El Baz, *A computational experience with distributed asynchronous iterative methods for convex network flow problems*, in Proc. the 28th IEEE Conference on Decision and Control, Tampa, FL, 1989, pp. 590–591.
24. B. Feijoo and R. R. Meyer, *Piecewise-linear approximation methods for nonseparable convex optimization*, Management Sci., 34 (1988), pp. 411–419.
25. J.-Ch. Fiorot and P. Huard, *Composition and union of general algorithms of optimization*, Math. Programming Study, 10 (1979), pp. 69–85.
26. H. Frank and W. Chou, *Routing in computer networks*, Networks, 1 (1971), pp. 99–122.
27. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, Cambridge, MA, 1994. Also available in html form on the Internet, URL `http://www.netlib.org/pvm3/book/pvm-book.html`.
28. G. Golub and J. M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, San Diego, CA, 1993.
29. A. Greenbaum, *Synchronization costs on multiprocessors*, Parallel Comput., 10 (1989), pp. 3–14.
30. J. L. Gustafson, G. R. Montry, and R. E. Banner, *Development of parallel methods for a 1024-processor hypercube*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 609–638.
31. P. T. Harker and J.-S. Pang, *Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications*, Mathematical Programming, 48 (1990), pp. 161–220.
32. J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, Springer-Verlag, Berlin, 1993.
33. R. W. Hockney and C. R. Jesshope, *Parallel Computers 2: Architecture, Programming and Algorithms*, Adam Hilger, Bristol, 1988.
34. K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, Singapore, 1985.
35. H. T. Kung, *Synchronized and asynchronous parallel algorithms for multiprocessors*, in Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, ed., Academic Press, New York, NY, 1976, pp. 153–200.
36. T. Larsson, A. Migdalas, and M. Patriksson, *A partial linearization method for the traffic assignment problem*, Optimization, 28 (1993), pp. 47–61.
37. T. Larsson and M. Patriksson, *A class of gap functions for variational inequalities*, Math. Programming, 64 (1994), pp. 53–79.
38. T. Larsson and M. Patriksson, *Equilibrium characterizations of solutions to side constrained asymmetric traffic assignment models*, Le Matematiche, 49 (1994), pp. 249–280.

39. T. Larsson and M. Patriksson, *Price-directive traffic management: applications of side constrained traffic equilibrium models*, report, Department of Mathematics, Linköping University, Linköping, Sweden, 1996.
40. L. S. Lasdon, *Optimization Theory for Large Systems*, MacMillan, New York, NY, 1970.
41. F. A. Lootsma and K. M. Ragsdell, *State-of-the-art in parallel nonlinear optimization*, Parallel Comput., 6 (1988), pp. 133–155.
42. D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, second ed., 1984.
43. Z.-Q. Luo and P. Tseng, *Error bounds and convergence analysis of feasible descent methods: a general approach*, Ann. Oper. Res., 46 (1993), pp. 157–178.
44. G. G. L. Meyer, *Asymptotic properties of sequences iteratively generated by point-to-set maps*, Math. Programming Study, 10 (1979), pp. 115–127.
45. J. C. Miellou, *Itérations chaotiques à retards: études de la convergence dans le cas d'espaces partiellment ordonnés*, Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris), Série A, 280 (1975), pp. 233–236.
46. J.-C. Miellou, P. Cortey-Dumont, and M. Boulbrachêne, *Perturbation of fixed-point iterative methods*, in Advances in Parallel Computing, Volume 1, D. J. Evans (ed.), JAI Press, Greenwich, CT, 1990, pp. 81–122.
47. A. Migdalas, *Cyclic linearization vs. Frank–Wolfe decomposition for nonlinear problems over Cartesian product sets*, unpublished note, Department of Mathematics, Linköping University, Linköping, Sweden, 1990.
48. A. Nagurney, *Network Economics: A Variational Inequality Approach*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.
49. S. S. Nielsen and S. A. Zenios, *Massively parallel algorithms for singly constrained convex programs*, ORSA J. Comput., 4 (1992), pp. 166–181.
50. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, 1970.
51. P. M. Pardalos, A. T. Phillips, and J. B. Rosen, *Topics in Parallel Computing in Mathematical Programming*, vol. 2 of Applied Discrete Mathematics and Theoretical Computer Science Series, Science Press, New York, NY, 1992.
52. M. Patriksson, *Partial linearization methods in nonlinear programming*, J. Optim. Theory Appl., 78 (1993), pp. 227–246.
53. M. Patriksson, *A unified description of iterative algorithms for traffic equilibria*, European J. Oper. Res., 71 (1993), pp. 154–176.
54. M. Patriksson, *A unified framework of descent algorithms for nonlinear programs and variational inequalities*, Doctoral dissertation, Department of Mathematics, Linköping University, Linköping, Sweden, 1993.
55. M. Patriksson, *Cost approximation: A unified framework of descent algorithms for nonlinear programs*, report, Department of Mathematics, University of Washington, Seattle, WA, 1994. Revised for possible publication in SIAM J. Optim.
56. M. Patriksson, *The Traffic Assignment Problem: Models and Methods*, VSP, Utrecht, 1994.
57. M. Patriksson, *On the convergence of descent methods for monotone variational inequalities*, Oper. Res. Lett., 16 (1994), pp. 265–269.
58. M. Patriksson, *A taxonomy of classes of descent algorithms for nonlinear programs and variational inequalities*, report, Department of Mathematics, University of Washington, Seattle, WA, 1994.
59. E. R. Petersen, *A primal-dual traffic assignment algorithm*, Management Sci., 22 (1975), pp. 87–95.
60. M. Ç. Pinar and S. A. Zenios, *Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm*, ORSA J. Comput., 4 (1992), pp. 235–249.
61. R. T. Rockafellar, *Local boundedness of nonlinear, monotone operators*, Michigan Math. J., 16 (1969), pp. 397–407.
62. R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
63. R. T. Rockafellar, *On the maximality of sums of nonlinear monotone operators*, Trans. Amer. Math. Soc., 149 (1970), pp. 75–88.
64. A. Ruszczyński, *An augmented Lagrangian decomposition method for block diagonal linear programming problems*, Oper. Res. Lett., 8 (1989), pp. 287–294.
65. G. L. Schultz and R. R. Meyer, *An interior point method for block angular optimization*, SIAM J. Optim., 1 (1992), pp. 583–602.
66. P. Tseng, *Dual ascent methods for problems with strictly convex costs and linear constraints: a unified approach*, SIAM J. Control Optim., 28 (1990), pp. 214–242.
67. P. Tseng, *Decomposition algorithm for convex differentiable minimization*, J. Optim. Theory Appl., 70 (1991), pp. 109–135.
68. P. Tseng, *On the rate of convergence of a partially asynchronous gradient projection algorithm*, SIAM J. Optim., 1 (1992), pp. 603–619.
69. P. Tseng and D. P. Bertsekas, *Relaxation methods for problems with strictly convex costs and linear constraints*, Math. Oper. Res., 16 (1991), pp. 462–481.

70. P. Tseng, D. P. Bertsekas, and J. N. Tsitsiklis, *Partially asynchronous, parallel algorithms for network flow and other problems*, SIAM J. Control Optim., 28 (1990), pp. 678–710.
71. J. N. Tsitsiklis and D. P. Bertsekas, *Distributed asynchronous optimal routing in data networks*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 325–332.
72. A. Üresin and M. Dubois, *Sufficient conditions for the convergence of asynchronous iterations*, Parallel Comput., 10 (1989), pp. 83–92.
73. A. Üresin and M. Dubois, *Asynchronous iterative algorithms: models and convergence*, in Advances in Parallel Algorithms, L. Kronsjö and D. Shumsheruddin (eds.), Blackwell Scientific Publications, Oxford, 1992, ch. 10, pp. 302–342.
74. J. van Tiel, *Convex Analysis: An Introductory Text*, John Wiley & Sons, Chichester, U.K., 1984.
75. N. Zadeh, *A note on the cyclic coordinate ascent method*, Management Sci., 16 (1969/1970), pp. 642–644.
76. W. I. Zangwill, *Nonlinear Programming: A Unified Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1969.
77. E. Zeidler, *Nonlinear Functional Analysis and its Applications II/B: Nonlinear Monotone Operators*, Springer-Verlag, New York, NY, 1990.

Received Date
Accepted Date
Final Manuscript Date