9

PARALLEL COST APPROXIMATION ALGORITHMS FOR DIFFERENTIABLE OPTIMIZATION

Michael Patriksson

Department of Mathematics, Box 354350
University of Washington, Seattle, Washington 98195-4350

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 encompasses the classical Gauss–Seidel scheme, a synchronized parallel algorithm which encompasses the Jacobi method, and a partially asynchronous parallel algorithm. The analysis validates inexact computations in both the subproblem and line search phases. 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have a direct correspondence to the increasing degree of parallelism and asynchronism, and the resulting usage of more outdated information in the algorithms.

Key words: Cartesian product sets, convex optimization, 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 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 different 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 on convex product sets, and study its 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 hence supplies some information about what can be expected when parallelizing an existing serial CA code for the problem under consideration.

1.1 A Differentiable Optimization Problem

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

$$X = \prod_{i \in \mathcal{C}} X_i, \qquad X_i \subseteq \Re^{n_i}, \qquad \sum_{i \in \mathcal{C}} n_i = n, \tag{9.1}$$

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). The problem under consideration is the constrained differentiable problem

$$\text{[CDP]} \qquad \min_{x \in X} f(x).$$

One example of a constraint structure of the form (8.1) is that of (generalized) box constraints,

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

unconstrained optimization is included in this framework. The structure (8.1) of the constraints of [CDP] is abundant in network programming (in which case the index set $\mathcal{C}$ usually denotes a set of commodities of goods or groups of vehicles in a transportation network), for example in market and traffic equilibrium models [46, 55]. Constraints of the form (8.1) are also obtained in subproblems in relaxation schemes for optimization problems with *block-angular* constraint structures, that is, where, in addition to the constraints defining (8.1), coupling constraints are present [20, 57], and in Lagrangean dual formulations of strictly convex programs [11, 65, 68]. We discuss two applications below.

1.2 Applications

Traffic assignment

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 (p, q) of origins and destinations of trips and their respective total demand for transportation, d_{pq}. Denoting by h_{pqr}, $r \in \mathcal{R}_{pq}$, the flow on route r between nodes p and q, the total flow on a link $a \in \mathcal{A}$ is given by

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

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 travel cost function $c_a : \Re_+ \mapsto \Re_{++}$. The *traffic assignment problem* (or, the *traffic equilibrium problem*) is to find a set of consistent route and link flows solving

$$\text{[TAP]} \qquad \min f(x) = \sum_{a\in\mathcal{A}} \int_0^{x_a} c_a(s)\, ds,$$

subject to

$$\sum_{r\in\mathcal{R}_{pq}} h_{pqr} = d_{pq}, \qquad \forall (p,q) \in \mathcal{C},$$

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

Disregarding the auxiliary link flow defining constraints, the problem [TAP] is clearly of the form [CDP]. (A problem with an identical structure arises in message routing problems in computer communication networks [26]. For a derivation and overview of [TAP], see [55].) To give an idea of the size of a practical problem, we mention that in one application [21] the number of nodes and links are 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, are problem sizes of the order of 1000, 3000 and $10,000$ in the number of nodes, links and commodities, respectively.

Separable strictly convex programming

The next example illustrates that although a problem may not have a Cartesian product structure, it may have a structure which can be exploited to enhance parallelization through a problem transformation. Let $f(x) = \sum_{i\in\mathcal{C}} f_i(x_i)$, $x\in\Re^n$, where each function $f_i : \Re^{n_i} \mapsto \Re$ is strictly convex, and consider the problem

[SC] $$\min_{x\in X} f(x),$$

where

$$X = \{\, x\in\Re^n \mid Ax = b; \quad x_i\in X_i,\ i\in\mathcal{C}\,\}.$$

We assume that X is nonempty and further, for simplicity, that the sets X_i are bounded. In the absence of the linear constraints, [SC] would separate into $|\mathcal{C}|$ problems solvable in parallel. We introduce a vector π of Lagrange multipliers for these constraints, and obtain the dual problem

[DSC] $$\max_{\pi}\ \theta(\pi) = \sum_{i\in\mathcal{C}}\theta_i(\pi) - b^{\mathrm{T}}\pi,$$

where

$$\theta_i(\pi) = \min_{x_i\in X_i}\left\{f_i(x_i) + \pi^{\mathrm{T}}A_ix_i\right\}, \qquad i\in\mathcal{C}, \tag{9.2}$$

A_i being the submatrix formed by the columns of A associated with x_i. The computation of $\theta(\pi)$ and $\nabla\theta(\pi)$ separates into $|\mathcal{C}|$ independent problems that may be solved in parallel, and, further, [DSC] is a problem of the form [CDP]; the primal optimal solution is obtained as the unique solution to (8.2) for any optimal dual vector π^*.

Remark 1 If f is nonseparable, then the primal subproblem is of the form [CDP], and the problem [SC] is an example of a block-angular program. In the absence of strict convexity, differentiability of the dual function may be introduced through the addition of a strictly convex penalty function for the equality constraints (e.g., as in augmented Lagrangean algorithms [31, 58, 62, 4]); thus, the problem is iteratively transformed into problems of the form [CDP]. The idea of utilizing Lagrange duality as a means to decomposition of a mathematical program was formulated at least as early as 1960 ([19]; see also [38]). Several transformations that enhance parallelism can be found in [11, Sec. 3.4]; most of them are based on Lagrange duality, and they all result in a problem of the form [CDP]. We note that more general constraints (convex inequalities) can be handled in the same way.

An example of a problem of the form [SC] is the strictly convex single-commodity network flow problem (e.g., [63, 20]), where A is the node-link incidence matrix for the network $\mathcal{G}$ and each set X_i is a box; this problem in turn includes the constrained matrix problem [12], the trip distribution problem in transportation analysis [55, Sec. 1.5.2], the quadratic transportation problem [20, 77], and several others.

Notation

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 (8.1), 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_- and i_+ are empty for $i = 1$ and $i = |\mathcal{C}|$, respectively.) Thus, we introduce the partitioning $x^{\mathrm{T}} = (x_1^{\mathrm{T}}, \ldots, x_{|\mathcal{C}|}^{\mathrm{T}})$, 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 components x_i. A 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

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

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 the set

$$L(x^0) = \{\, x \in X \mid f(x) \leq f(x^0) \,\}.$$

1.3 The Cost Approximation Algorithm

Derivation

The iterative algorithms studied in this paper are applications to [CDP] of the class of cost approximation (CA) algorithms, introduced and analyzed in [50, 51, 52, 53]. These algorithms are designed to find stationary points of [CDP], that is, solutions $x^* \in X$ to the *variational inequality problem*

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

We let Ω denote the set of stationary points of [CDP]. (A stationary point is a global minimizer of f whenever f is pseudoconvex [3, Sec. 3.5].)

A CA algorithm alternates between the following two steps:

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

(2) 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 subproblem 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

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

We let $Y(x)$ denote the solution set of $[\mathrm{VIP}_\Phi]$. (This set may also be characterized as follows: letting $N_X(x)$ be the *normal cone* to X at x, that is, $N_X(x) = \{\, y \in \Re^n \mid y^{\mathrm{T}}(z - x) \leq 0,\ \forall z \in X \,\}$ for $x \in X$ and $N_X(x) = \emptyset$ for $x \notin X$, and Π^{-1} denote the inverse mapping of Π, $Y(x) = [\Phi + N_X]^{-1}[\Phi - \nabla f](x)$, $x \in X$. This form is sometimes preferable to that of $[\mathrm{VIP}_\Phi]$, for example for recognizing certain instances of CA algorithms, like *operator splitting* algorithms [22], but also in the analysis of the properties of the CA subproblem.)

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

$$[\mathrm{CDP}_\varphi] \qquad \min_{y \in X} f_\varphi(y) = \varphi(y) + f(x) - \varphi(x) + [\nabla f(x) - \nabla \varphi(x)]^{\mathrm{T}}(y - x).$$

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 $[\mathrm{CDP}_\varphi]$.

The CA algorithm is adapted to the structure of [CDP] by defining the mappings Φ to be separable with respect to the partition of $\Re^n$ given by (8.1), that is, by letting

$$\Phi(x)^{\mathrm{T}} = [\Phi_1(x_1)^{\mathrm{T}}, \ldots, \Phi_{|\mathcal{C}|}(x_{|\mathcal{C}|})^{\mathrm{T}}], \qquad x \in X, \tag{9.3}$$

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

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

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

$$[\mathrm{CDP}_{\varphi_i}] \qquad \min_{y_i \in X_i} f_{\varphi_i}(y_i) = \varphi_i(y_i) + [\nabla_i f(x) - \nabla \varphi_i(x_i)]^{\mathrm{T}} y_i.$$

In the convergence analysis in [50, 51, 52, 53] of the CA algorithm, two forms of the cost approximating mapping have been used. The most general case is the choice of 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. In the less general case, the mappings are assumed to vary continuously with the iterates, that is, it is assumed that 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 $\{\Phi^t\}$ has the form $\Phi(\cdot, x^t)$.

Properties of the search directions

Before giving concrete examples of CA algorithms, we will provide some important properties of the subproblems $[\mathrm{VIP}_{\Phi}]$ and $[\mathrm{CDP}_{\varphi}]$, which we will frequently be referring to in the subsequent analysis. As with all the results presented in this paper, we will give a sketch of a proof, and refer the reader to a complete proof.

Lemma 1 [53](Properties of the subproblem and search direction) *Let $x \in X$, $Y(x)$ be the (possibly empty) set of solutions to* $[\mathrm{VIP}_{\Phi}]$, *$y \in Y(x)$, and $d = y - x$.*

(a) (A fixed point characterization of Ω)

- **(1)** *$x \in \Omega \iff x \in Y(x)$.*
- **(2)** *Let $\Phi \equiv \nabla \varphi$. Then, $x \in \Omega \iff x \in Y(x) \iff f_{\varphi}(y) = f_{\varphi}(x)$.*

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

- **(1)** *If Φ is strictly monotone on X, then $\nabla f(x)^{\mathrm{T}} d < 0$.*
- **(2)** *Let $\Phi \equiv \nabla \varphi$. Let $\overline{y} \in X$ be any point such that $f_{\varphi}(\overline{y}) < f_{\varphi}(x)$, and $\overline{d} = \overline{y} - x$. Then, $\nabla f(x)^{\mathrm{T}} \overline{d} < 0$. Especially, $\nabla f(x)^{\mathrm{T}} d < 0$.*

(3) *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 $\nabla f(x)^{\mathrm{T}}\overline{d} < 0$ holds. Especially, $Y(x)$ is nonempty and singleton, and

$$\nabla f(x)^{\mathrm{T}} d \leq -m_\Phi \|d\|^2. \tag{9.4}$$

(c) (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 mapping $x \mapsto D(x) = Y(x) - x$ is closed on X.*

Sketch of proof.

(a) **(1)** Follows directly from the construction of [VIP$_\Phi$].

(2) The first equivalence is a special case of **(1)**; the second follows immediately from the construction of [CDP$_\varphi$].

(b) **(1)** The result **(a)(1)** implies that $d \neq 0$. Combine [VIP$_\Phi$] with the strict monotonicity of Φ.

(2) Combine the inequality with the convexity of φ. By **(a)(2)**, y is a special case of such a $\overline{y}$.

(3) Similarly to the proof of **(1)**, using the strong monotonicity of Φ we obtain that $\nabla f(x)^{\mathrm{T}}\overline{d} \leq \varepsilon - m_\Phi \|\overline{d}\|^2$. The second result then follows from letting $\varepsilon = 0$, and the properties of $Y(x)$ follow by the strong monotonicity of Φ.

(c) By the continuity of ∇f and Φ, the mapping $x \mapsto Y(x)$ is closed on X. The search direction mapping is an affine transformation of this mapping, and is therefore also closed on X. □

Remark 2 The fixed point property provides a termination criterion for the algorithm. The fixed point mapping $x \mapsto Y(x)$ is a point–to–set mapping in general, unless Φ is strictly monotone on X. 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 **(c)**. A sufficient condition for the set $Y(x)$ to be both nonempty and bounded for every $x \in X$ is that Φ is *coercive* [76], that is, that either X is bounded or

$$\lim_{\substack{\|x\| \to +\infty \\ x \in X}} \frac{\Phi(x)^{\mathrm{T}}(x - z)}{\|x\|} = +\infty, \tag{9.5}$$

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

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

necessary and sufficient conditions are found in [53].

Step length rules

The convergence analysis reveals that the more general the form of the mappings Φ^t the stronger monotonicity properties will necessarily have to be imposed on them. (The choice of cost approximating mappings as gradients somewhat lessens the monotonicity requirements.) However, with stronger monotonicity properties, the flexibility with which step lengths in the second main stage of the algorithm may be chosen increases, and simultaneously the computational burden decreases. (This is important in applications where objective values are expensive to calculate.) We consider three rules for choosing step lengths; 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 will be 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*

$$\min \{ f(x + \ell d) \mid \ell \geq 0; \quad x + \ell d \in X \}$$

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

$$f(x + \beta^i d) - f(x) \leq \alpha \beta^i \nabla f(x)^{\mathrm{T}} d. \tag{9.6}$$

(R) (Relaxation step) *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 [53](Validation of step length rules) *Let $x \in X \setminus \Omega$, $y \in Y(x)$, and $d = y - x$.*

(A) **(1)** *Let Φ be strictly monotone on X. Then, the inequality (8.6) holds for a finite integer i.*

(2) *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 (8.6) holds for a finite integer i.*

(3) *Assume that ∇f is Lipschitz continuous on X. Then, either a unit step satisfies (8.6) or*

$$\beta^{\overline{i}} > 2\beta(\alpha - 1) \frac{\nabla f(x)^{\mathrm{T}} d}{M_{\nabla f} \|d\|^2}. \tag{9.7}$$

(R) *Assume that ∇f is Lipschitz continuous on X, and let Φ be strongly monotone on X. Then,*

$$f(x + \ell d) - f(x) \leq \ell(-m_\Phi + M_{\nabla f}/2\ell)\|d\|^2. \tag{9.8}$$

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

Sketch of proof.

(A) **(1)** Taylor's formula yields, for $\ell \in [0, 1]$,

$$f(x + \ell d) - f(x) = \ell \nabla f(x)^{\mathrm{T}} d + o(\ell). \tag{9.9}$$

For any value of $\alpha \in (0, 1)$, there must therefore be an $\overline{\ell} > 0$ such that, for all $\ell \in (0, \overline{\ell})$, $f(x + \ell d) - f(x) \leq -\alpha \ell \nabla f(x)^{\mathrm{T}} d$ holds, whence the result.

(2) Follows as in **(1)**, using Lemma 1.b.2.

(3) Using Taylor's formula, the Lipschitz continuity of ∇f yields, for all $\ell \in [0, 1]$,

$$f(x + \ell d) - f(x) \leq \ell \nabla f(x)^{\mathrm{T}} d + (M_{\nabla f}/2\ell^2)\|d\|^2. \tag{9.10}$$

Together with the descent property of d (cf. Lemma 1.b.1), this inequality yields that, by choosing

$$\ell \leq 2(\alpha - 1)\nabla f(x)^{\mathrm{T}} d/(M_{\nabla f}\|d\|^2), \tag{9.11}$$

$f(x + \ell d) - f(x) \leq \alpha \ell \nabla f(x)^{\mathrm{T}} d$ holds. With $\ell = \beta^{\bar{\imath}}$, $\bar{\imath}$ is the smallest nonnegative integer i to satisfy (8.11).

(R) The inequality (8.8) follows from (8.10) and (8.4). □

Note that the step lengths resulting from the use of rules A and R vary with the choice of Φ. For example, choosing Φ such that the value of m_Φ is large enough, both rules are valid using unit step lengths.

Instances of cost approximation algorithms

In Table 1, we present some instances of the cost approximation 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.

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	$\nabla f + (1/\gamma_t) r^t$

1.4 Preview

In the rest of the paper we will demonstrate how the structure of [CDP] can be exploited in CA algorithms, decomposition being enabled by choosing the cost approximating mappings separable with respect to the product set. 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 (8.1) is not the only possible—a set X_i may itself be a Cartesian product of convex sets. The number of parallel processors available dictates how many independent subproblems [VIP$_{\Phi_i}$] 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$_\Phi$] is more accurate; in the sequential version of the algorithm, this may be used to reach a certain level of accuracy in less iterations. In the extreme case, of course, we have only one component; we have then defined an instance of the original algorithm, and thus have no decomposition.) The choice of $\mathcal{C}$ may also aid in the construction of a partition which makes the most efficient use of the computer. An application is to the case where [VIP$_\Phi$] is too large to be mapped in its entirety onto the computer available; the choice of $\mathcal{C}$ may be tailored for the computer by defining a decomposition of [VIP$_\Phi$] 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 number of elements in $\mathcal{C}$ is of importance (e.g., Theorem 11).

A convergence analysis is made for three decomposition versions of the cost approximation algorithm. In the analysis, rather than focusing on the rate of convergence we concentrate on finding 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 encompasses 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, especially when the original problem is too large to be handled without decomposition, and also for parallel computers whenever not all the independent subproblems can be solved simultaneously. In the second version, which encompasses the classical Jacobi approach in unconstrained optimization, the independent subproblems defined by $\mathcal{C}$ 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 computations, as a means to circumvent 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, 71, 44, 72]); in our case, it would, for example, be possible to utilize the result of Lemma 1.a in such an analysis. We will, however, make use of the descent properties of the CA algorithm (Lemma 1.b).

Decomposition algorithms for [CDP] of these forms are not 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 [66, 67]. The theoretical results were first established in [54], where detailed proofs can be found; overviews of parallel differentiable optimization are found in [39, 11, 49].

The rest of the paper is organized as follows. In Section 2, we analyze the sequential, Gauss–Seidel type, approach. In Section 3, the Jacobi type, synchronized parallel algorithm is studied. In Section 4, we analyze the partially asynchronous version of this algorithm. In Section 5, we discuss some possible extensions, and, finally, Section 6 contains some exercises.

2 SEQUENTIAL COST APPROXIMATION ALGORITHMS

2.1 The Sequential Algorithm

Derivation

The sequential version of the algorithm is of interest perhaps primarily for applications to 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 $\mathcal{C}$.

Conceptually, 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 continuous and monotone cost approximating mapping $\Phi^t_{i_t}$, and solve the problem of finding $y^t_{i_t} \in X_{i_t}$ such that $(i = i_t)$

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

(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.

The Gauss–Seidel scheme

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

$$i_t = t \pmod{|\mathcal{C}|} + 1. \tag{9.12}$$

Choose the cost approximating mapping

$$\Phi^t_i = \nabla_i f(x^t_{i_-}, \cdot, x^t_{i_+}). \tag{9.13}$$

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 a convexity assumption on f with respect to each component x_i) to finding

$$y^t_i \in \arg\min_{y_i \in X_i} f(x^t_{i_-}, y_i, x^t_{i_+}). \tag{9.14}$$

An exact line search would always 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 [48] (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 (8.13) 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.

2.2 Convergence of the Cyclic Cost Approximation Algorithm

The conceptual algorithm

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

$$\varphi(y, x) = \sum_{i \in \mathcal{C}} \varphi_i(y_i, x), \qquad x, y \in X, \tag{9.15}$$

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, $\Phi_i^t \equiv \nabla_{x_i} \varphi(\cdot, x^t)$.] Further, we shall assume that the line search is exact (i.e., Rule E is used).

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 (8.15). In the sequential CA algorithm, let the sequence $\{i_t\}$ be chosen according to the cyclic rule (8.12), and Rule E be used. Assume that $x^0 \in X$ is such that the lower level set $L(x^0)$ is bounded. Assume 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*

$$\left\{ \inf_{x \in \Omega \cap L(x^0)} \|x^t - x\| \right\} \to 0. \tag{9.16}$$

Sketch of proof. The proof utilizes Theorem A of Zangwill [75, Sec. 4.5], identifying the solution set with Ω, the continuous descent function with f, and the algorithmic map with the composite map $A = E \circ D_{|\mathcal{C}|} \circ E \circ D_{|\mathcal{C}|-1} \circ \ldots \circ E \circ D_1$, where $x \mapsto D_i(x) = Y_i(x) - x_i$, $i \in \mathcal{C}$, is the search direction finding map, and E is the exact line search map. To fulfill the assumptions of Zangwill's Theorem, we need to show that (1) the sequence $\{x^t\}$ lies in a compact set, (2) 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 (3) 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 of the algorithm as given in Table 2.)

First, the set $L(x^0)$ is compact, by assumption and the continuity of f; the sequence $\{x^t\}$ lies in this set by the descent property. Second, d^t defines a descent direction by Lemma 1.b.1, and the descent property follows from the exact line search. If $x^t \in \Omega$, then by Lemma 1.a.2, $x \in Y_i(x)$, $i \in \mathcal{C}$, and the algorithm is terminated at Step 2. Third, let $i \in \mathcal{C}$. The closedness of the mapping Y_i, together with the boundedness assumption, can be used to show that the mapping Y_i is locally bounded; that is sufficient for the sequence $\{y^t\}$ to be bounded. The mapping E is closed on $L(x^0)$ since f is continuous; the composite map $E \circ D_i$ is therefore closed on $L(x^0)$. It is also compact valued, from the descent property and the compactness of $L(x^0)$. The algorithmic mapping A is therefore closed on $L(x^0)$.

Applying Zangwill's Theorem A, 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 $\{x^t\}$.) The property (8.16) follows from the boundedness of $L(x^0)$. □

For the unconstrained case, a convergence result for the Gauss–Seidel algorithm corresponding to that special case of the above theorem is given by Zadeh [74]; see also [11, Prop. 3.2.5] and [48, Th. 14.6.7].

A generalization of Zangwill's classical result will enable us to weaken the assumption that the subproblems [CDP$_{\varphi_i}$] are solved exactly, and also allows for the use of more general rules for choosing indices. The technique used is based on the *spacer step* theorem [40, 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

$$C(x) = \{\, y \in X \mid f(y) \leq f(x) \,\}, \qquad x \in X; \tag{9.17}$$

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

Truncated cyclic cost approximation

Although each subproblem [CDP$_{\varphi_i}$] may be of much smaller dimension than the original problem, it may still be computationally demanding to solve, and in practice it may not be possible to solve exactly. Motivated by these practical considerations, we will introduce a truncated version of the sequential CA algorithm in which the solution of [CDP$_{\varphi_i}$] is terminated finitely. We will assume that the algorithm employed for solving [CDP$_{\varphi_i}$] and the termination criteria used satisfy the following assumption.

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.*

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

(2) (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 $(y_i, x) \in B_i(z_i, x)$.*

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

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

Remark 3 The condition **(1)** ensures that the fixed point property of the original algorithm is preserved, and that the algorithm terminates whenever a solution to [CDP$_{\varphi_i}$] has been obtained. The condition **(2)**, together with the lower bound in **(4)** on the number of iterations performed and Lemma 1.b.2, ensures that the resulting approximate solution $\overline{y}_i$ of [CDP$_{\varphi_i}$] defines a direction of descent with respect to f. Condition **(3)** 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 **(4)**, finally, limits the work done in each main iteration of the cyclic CA algorithm. Assuming that X is bounded, 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.

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 $1 \leq k^t_{i_t} \leq \overline{k}_i$ iterations of the algorithm described by B_{i_t}, starting from $x^t_{i_t}$, that is, let

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

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

Note that the value of $k^t_{i_t}$ need not be determined *a priori*, and should 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 cost approximation 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 cost approximation algorithm.*

Sketch of proof. We begin by identifying the algorithmic map A. We assume that $\{x^t\}$ is infinite (otherwise, by Assumption 1.1, 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 1.4 and the use of the cyclic rule imply that $\{k^t_i\}_{\mathcal{T}^i}$ 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 $\{k^t_i\}_{\mathcal{T}^i}$. 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 A is given by

$$A = E \circ \overline{D}_{|\mathcal{C}|} \circ E \circ \overline{D}_{|\mathcal{C}|-1} \circ \ldots \circ E \circ \overline{D}_1,$$

where $\overline{D}_i(x) = \overline{Y}_i(x) - x_i$.

The conditions of Zangwill's Theorem are satisfied for A, by arguments similar to those in the proof of Theorem 1. [The u.s.c. and compactness assumptions on B_i are required since B_i may lead outside of the compact set $L(x^0)$.]

We then 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 (8.17). Hence, all the conditions of the Spacer Step Theorem [40, p. 231] are fulfilled. The rest of the proof follows as in that of Theorem 1. □

In [51] it was shown that although the conditions on the algorithm used for solving [CDP$_{\varphi_i}$] are, apparently, fairly mild, the closedness property implies that the sequence $\{\overline{y}_i^t\}$ of approximate solutions is optimizing in [CDP$_{\varphi_i}$] in the limit.

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, that is,

$$\mathcal{C} \subseteq \{i_t, i_{t+1}, \ldots, i_{t+N-1}\}, \qquad \forall t \geq 1. \tag{9.18}$$

Within the limits defined by the value of N, one may use any strategy for choosing indices, and (8.18) 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 (8.12) by the essentially cyclic rule (8.18) in the cyclic cost approximation algorithm. Then the conclusions of Theorem 1 holds for the essentially cyclic cost approximation algorithm.*

Sketch of 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. the essentially cyclic rule), 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 (8.17). We can therefore invoke the Spacer Step Theorem, and the rest of the proof follows that of Theorem 1. $\square$

Combining the two maps A used in the proofs of Theorems 2 and 3, one may establish the convergence of a *truncated essentially cyclic CA algorithm.*

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 (8.6), different values of α and β may be chosen for the different components.)

The sequential cost approximation algorithm using the Armijo step length rule will be established (under the essentially cyclic rule) for strictly monotone mappings Φ of the form

$$\Phi(y, x)^{\mathrm{T}} = [\Phi_1(y_1, x)^{\mathrm{T}}, \ldots, \Phi_{|\mathcal{C}|}(y_{|\mathcal{C}|}, x)^{\mathrm{T}}], \qquad x, y \in X; \tag{9.19}$$

to ensure that the subproblem solution is locally bounded, we also impose the condition that

$$\Phi_i(x_i, x) - \nabla_i f(x) \in \mathtt{int}\,(\mathtt{dom}\,([\Phi_i(\cdot, x) + N_{X_i}]^{-1})), \qquad x \in L(x^0); \tag{9.20}$$

it is automatically satisfied when [cf. (8.5)] X_i being bounded or

$$\lim_{\substack{\|y_i\| \to +\infty \\ y_i \in X_i}} \frac{\Phi_i(y_i, x)^{\mathrm{T}}(y_i - z_i)}{\|y_i\|} = +\infty, \qquad y \in X, \tag{9.21}$$

for some $z_i \in X_i$.

Theorem 4 (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 (8.19), 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 (8.18), 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 (8.20) 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 (8.16) holds.*

Sketch of proof. The sequence $\{x^t\}$ is bounded, by the boundedness of $L(x^0)$ and the descent property. The conditions on $\{\Phi^t\}$ imply that the mapping $x \mapsto Y_i(x)$ is locally bounded on X, and therefore also the sequence $\{y^t\}$ is bounded. Let x^∞ and y^∞ be arbitrary accumulation points of $\{x^t\}$ and $\{y^t\}$, respectively, which we, for simplicity, will associate with the same convergent subsequence $\mathcal{T}$.

Having showed that $\{d^t\}$ is bounded, to show that actually $\{d^t\} \to 0$ holds, we argue by contradiction. By Lemma 2.A.1, the characterization $[\mathrm{VIP}_\Phi]$ of y^t, and the strict monotonicity of $\Phi_i(\cdot, x^t)$,

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

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 $\overline{t}$ such that for every $t \geq \overline{t}$ in $\overline{\mathcal{T}}$, the step length produced by Rule A is less than one, that is,

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

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 $\{\nabla f(x^t)^{\mathrm{T}} d^t\} \to 0$ 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 $[\mathrm{VIP}_{\Phi_i^t}]$, using the strict monotonicity of Φ_i we then reach a contradiction. Thus, $y^\infty = x^\infty$ must actually hold, and hence, $\{d^t\} \to 0$.

From the subsequence $\mathcal{T}$ we extract a further 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,

$$\|x^{t+1} - x^t\| = \beta^{\overline{\imath}_t} \|d^t\| \leq \|d^t\|,$$

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

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

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

$$\nabla_{\imath_j} f(x^\infty)^{\mathrm{T}}(x_{\imath_j} - x_{\imath_j}^\infty) \geq 0, \qquad \forall x_{\imath_j} \in X_{\imath_j} \tag{9.22}$$

holds for $j = 0, 1, \ldots, N-1$. Since $\mathcal{C} \subseteq \{\imath_0, \imath_1, \ldots, \imath_{N-1}\}$, it follows that

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

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 also assuming only monotonicity of $\Phi_i(\cdot, x)$ if it is a gradient mapping.) To ensure that it holds also for the remaining indices, the requirement (8.18) is necessary, however. This result complies with those obtained for fixed point algorithms using the *free-steering* order (e.g., [48, 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 8). Also, in the basic 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 9). 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 [66] establishes a convergence result similar to Theorem 4 for the nonlinear proximal descent algorithm (an instance of the CA algorithm; see [51] and [52, Th. 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 cost approximation algorithm for strongly monotone cost approximation 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 maximal 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 5 (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 (8.18). 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, (8.16) holds. If further Ω is finite, then for the Rules E, A and R, $\{x^t\}$ converges to a point in Ω.

Sketch of proof. Similarly to the proof of Theorem 4 for Rule A, utilizing Lemmas 1.b.3 and 2.A.1,

$$f(x^{t+1}) - f(x^t) \le \alpha \ell_t \nabla f(x^t)^{\mathrm{T}} d^t \le -\alpha \ell_t m_{\Phi_{i_t}^t} \|d^t\|^2, \qquad t = 0, 1, \ldots \tag{9.23}$$

holds. Since f is lower bounded, it follows as in that proof that $\{d^t\} \to 0$. Since the left–hand–side of the first inequality in (8.23) 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

$$f(x^{t+1}) - f(x^t) \le -\rho \|d^t\|^2 \tag{9.24}$$

holds, with $\rho = \varepsilon^2 M_{\nabla_i f}/2$. Since f is lower bounded on X, (8.24) 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 $\delta > 0$, $\|x^{t+1} - x^t\| \leq \delta\|d^t\|$ holds. Constructing a convergent subsequence $\overline{\mathcal{T}}$ in a manner similar to that in the proof of Theorem 4, we may then establish that $\{x^{t+j}\}_{t\in\overline{\mathcal{T}}} \to x^\infty$ and $\{y^{t+j}\}_{t\in\overline{\mathcal{T}}} \to x^\infty$ holds for $j = 0, 1, \ldots, N-1$. The points $y_{\imath_j}^{t+j}$, $j = 0, 1, \ldots, N-1$, are characterized by

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

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

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

for $j = 0, 1, \ldots, N-1$. In the limit of $t \in \overline{\mathcal{T}}$, this inequality implies that (8.22) holds for $j = 0, 1, \ldots, N-1$, that is, $x^\infty \in \Omega$. The result (8.16) follows as in the proof of Theorem 1; the last result follows from a known result on bounded sequences $\{x^t\}$ satisfying $\{x^{t+1} - x^t\} \to 0$ [48, Th. 14.1.5]. □

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 with respect to 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.

Cohen [16, 17] establishes a similar result for the auxiliary problem principle (an instance of the CA algorithm [52, Th 4.1]) in the case of Rule P with $\ell_t^i \equiv 1$; 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 [66] 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 described by a continuous mapping on $X \times X$.

One instance of the cyclic cost approximation algorithm using Rule P 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. (See also Table 1.)

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; the convergence characteristics of the algorithm is thus for the most cases already known. We discuss implementational aspects of the algorithm, show that the classical Jacobi algorithm is a special case, and identify drawbacks originating from the enforced synchronization; the latter motivates the introduction of the asynchronous parallel algorithm in Section 4.

3.1 Synchronized Parallel Computations

We concentrate on the parallelism inherent in the subproblems of the CA algorithm, although there may be parallelism involved on 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 the problem [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 the dynamic allocation strategy improves the load balance among the processors [18]; this is done at the expense of the overhead caused by the handling of the task queue [72]. (The subproblems in parallel CA algorithms for [TAP] may have different and unpredictable complexity, due to the very different spatial separations of the different O-D pairs; the load balance may, however, be improved through the proper choices of the partition defined by $\mathcal{C}$ and of the stopping criteria in truncated subproblems algorithms. In contrast, the Lagrange subproblems in quadratic transportation problems have closed form solutions.)

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, that is, 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 algorithms in Section 4.

3.2 The Synchronized Parallel Algorithm

Derivation

We let the partition of $\Re^n$ be defined by (8.1). In the original CA algorithm, given an iteration t and a point $x^t \in X$, let the monotone cost approximating mapping Φ^t be of the form (8.3). It then follows that the subproblem $[\mathrm{VIP}_{\Phi^t}]$ separates into $|\mathcal{C}|$ independent subproblems $[\mathrm{VIP}_{\Phi_i^t}]$, 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 matches the separability structure. This also means that the convergence results for the parallel algorithm are identical to those of the original algorithm for the particular choices of cost approximating mappings.

A summary of the synchronized parallel CA algorithm is given in Table 3. The algorithms presented in Table 1 all have natural parallel decomposition versions.

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 $[\mathrm{VIP}_{\Phi_i^t}]$ 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 $[\mathrm{VIP}_{\Phi_i^t}]$ 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 Jacobi scheme

In the parallel algorithm, choose the cost approximating mapping of the form (8.3), where

$$\Phi_i^t = \nabla_i f(x_{i_-}^t, \cdot, x_{i_+}^t), \qquad i \in \mathcal{C}. \tag{9.25}$$

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 [48, 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 (8.25) and unit steps are used.

Each component of the mapping Φ^t is of the form (8.13), 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, Chapter 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 [51] and [55, 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., [47]).

3.3 Convergence of the Synchronized Parallel Cost Approximation Algorithm

The conceptual algorithm

In the convergence result corresponding to Theorems 1 and 3, 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 (8.15); the convergence conditions are identical.

Theorem 6 (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 (8.15), 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 [CDP_{φ_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 (8.16) holds.*

Sketch of proof. The proof relies on Zangwill's Theorem, and follows very closely that of Theorem 1; here, the algorithmic map A has the form

$$A = E \circ \prod_{i \in \mathcal{C}} D_i,$$

where $D_i(x) = Y_i(x) - x_i$, $i \in \mathcal{C}$. □

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:

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 value of k_i^t is allowed to vary from one iteration to the next, it 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 by other criteria.

The following convergence result corresponds to Theorem 2, and has the same conditions.

Theorem 7 (Convergence of the truncated synchronized parallel cost approximation algorithm) *Replace Step 1 by Step 1" in the synchronized parallel CA* *algorithm. In Theorem 6, 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 6 hold for the truncated synchronized parallel CA algorithm.*

Sketch of proof. The result is established by the use of the Spacer Step Theorem. Similarly to the proof of Theorem 2, we may 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 A is given by

$$A = E \circ \prod_{i \in \mathcal{C}} \overline{D}_i,$$

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. □

Different step length rules

We next replace the exact line search Rule E by the Armijo step length Rule A.

Theorem 8 (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 (8.19), 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 (8.15), 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* $[\mathrm{VIP}_{\Phi_i}]$ *(respectively,* $[\mathrm{CDP}_{\varphi_i}]$*) is well defined, in the sense that (8.20) 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 (8.16) holds.*

Sketch of proof. Similarly to the proof of Theorem 4, 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 $[\mathrm{VIP}^t_\Phi]$ implies that $x^\infty \in \Omega$. For the other alternative, we obtain, by Lemma 1.c, that in the limit of $[\mathrm{VIP}^t_{\nabla\varphi}]$, $y^\infty \in Y(x^\infty)$ 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.1, $x^\infty \in \Omega$. The rest of the proof follows that of Theorem 1. □

We note that the Lipschitz continuity assumption here is stronger than that in Theorem 4 for the corresponding sequential algorithm, the reason being that the Armijo step length rule here is used over all components of x simultaneously. On the other hand, 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.

Theorem 9 (Convergence of the truncated synchronized parallel CA algorithm under Rule A) *In the truncated synchronized parallel CA algorithm of Theorem 7, let Rule A be used. Assume that the sequence $\{\overline{y}^t\}$ of truncated subproblem solutions is bounded. Then, the conclusions of Theorem 7 hold for the truncated synchronized parallel CA algorithm under the Armijo step length rule.*

Sketch of proof. The Armijo rule is valid (cf. Lemma 1.b.2, Lemma 2.a.2 and Assumption 1.1). 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, $f_\varphi(\overline{y}^\infty) \leq f_\varphi(x^\infty)$ 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

$$f_\varphi(\overline{y}^\infty) = f_\varphi(x^\infty). \tag{9.26}$$

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 $[\mathrm{CDP}^t_{\varphi_i}]$. In the limit of this subsequence, $[\mathrm{CDP}^t_{\varphi_i}]$ is solved using k_i steps of the algorithm B_i. By arguments similar to those in the proof of Theorem 1, 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 (8.26), Lemma 1.b.2 and Assumption 1.2, 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, that is however not a necessary property); a sufficient condition is, however, that X is bounded. Also note that the proof establishes that $\overline{y}^\infty \in Y(x^\infty)$, that is, the truncated algorithm is optimizing in the subproblem in the limit.

The next result corresponds to Theorem 5 for the sequential algorithm.

Theorem 10 (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 (8.3), 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, (8.16) holds. If further Ω is finite, then for the Rules E, A and R, $\{x^t\}$ converges to one point in Ω.

Sketch of proof. Similarly to the proof of Theorem 5, we obtain that $\{d^t\} \to 0$. From [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,

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

that is, $x^\infty \in \Omega$. The rest of the proof follows that of Theorem 5. □

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

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

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

$$\frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j^t}\}}{M_{\nabla f}} \leq \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}, \qquad i \in \mathcal{C}, \tag{9.27}$$

and we conclude that, in general, longer steps are allowed in the sequential algorithm than in the synchronized parallel algorithm. (A similar 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 well be more efficient in terms of solution time. The intuitive explanation for this observation is that the sequential algorithm utilizes the more recent information when constructing the next subproblem.

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, as a result, 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 much more complicated.) Recent numerical experiments indicate, however, that the introduction of such *asynchronous* computations can substantially enhance the efficiency of parallel iterative methods [23, 8, 13].

4.1 Asynchronous Parallel Computations

The convergence of asynchronous iterative methods have been studied by several researchers (e.g., [14, 43, 2, 6, 11, 71, 44, 72]). While *totally asynchronous*, or *chaotic*, algorithms allow arbitrarily large communication delays and differences in the frequency of computation of different processors, other algorithms, which are referred to as *partially asynchronous*, can not be established convergent unless an upper bound on these delays is imposed; the latter seems to be the case especially for gradient-type algorithms (see [11, Sec. 6.3.2]). Examples of totally asynchronous algorithms include dynamic programming methods, fixed point methods, and coordinate search methods for linear and convex network flows (e.g., [5, 6, 10, 9, 11, 7]). Partially asynchronous versions of instances of the CA algorithm have been studied only to a limited extent; variable metric (or, deflected gradient) methods in unconstrained optimization [11, Sec. 7.5], gradient projection methods [70, 11, 67], and coordinate ascent methods for dual formulations of strictly convex network flow problems [69] 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 $\mathcal{T}^i = \cup_{k \geq 0}\{k|\mathcal{C}| + i - 1\}$ 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 2 (Partial asynchronism) *There exists a positive integer P such that*

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

(2) $0 \leq t - \tau_j^i(t) \leq P-1$ *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, 69]; 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

$$x_i(t+1) = x_i(t) + \ell(y_i(t) - x_i(t)), \qquad t \in \mathcal{T}^i, \qquad i \in \mathcal{C}, \tag{9.28}$$

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

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

denote this vector.

4.2 Convergence of the Partially Asynchronous Parallel Cost Approximation Algorithm

The next result is an extension of Theorem 10 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 choose the sequence of cost approximating mappings as a continuous mapping on $X \times X$.)

Theorem 11 (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 (8.19), where $\Phi_i(\cdot, x)$*

is strongly monotone on X_i and Lipschitz continuous on X_i. Let Assumption 2 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

$$\ell \in \left(0, \frac{2\min_{i\in\mathcal{C}}\{m_{\Phi_i}\}}{M_{\nabla f}[1+(|\mathcal{C}|+1)P]}\right), \tag{9.29}$$

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 (8.16) holds. If further Ω is finite, then $\{x(t)\}$ converges to one point in Ω.

Sketch of proof. From the Lipschitz continuity of ∇f and the strong monotonicity of $\Phi_i(\cdot, x)$, $i \in \mathcal{C}$, we obtain

$$\begin{aligned}
f(x(t+1)) - f(x(t)) &\leq \ell \sum_{i\in\mathcal{C}} \nabla_i f(x(t))^{\mathrm{T}} d_i(t) + (M_{\nabla f}/2\ell^2)\|d(t)\|^2 \\
&= \ell \sum_{i\in\mathcal{C}} \nabla_i f(x^i(t))^{\mathrm{T}} d_i(t) + (M_{\nabla f}/2\ell^2)\|d(t)\|^2 \\
&\quad + \ell \sum_{i\in\mathcal{C}} \left[\nabla_i f(x(t)) - \nabla_i f(x^i(t))\right]^{\mathrm{T}} d_i(t) \qquad (9.30) \\
&\leq \ell\left(-\min_{i\in\mathcal{C}}\{m_{\Phi_i}\} + M_{\nabla f}/2\ell\right)\|d(t)\|^2 \\
&\quad + M_{\nabla f}\ell \sum_{i\in\mathcal{C}} \|d_i(t)\| \cdot \|x(t) - x^i(t)\|. \qquad (9.31)
\end{aligned}$$

Next, we bound $\|x(t) - x^i(t)\|$, $i \in \mathcal{C}$. From Assumption 2.2 and the use of Rule R, we have, for all $i, j \in \mathcal{C}$,

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

Using the triangle inequality, we obtain

$$\|x^i(t) - x(t)\| \leq \ell \sum_{\tau=t-P}^{t-1} \|d(\tau)\|, \qquad i \in \mathcal{C}. \tag{9.32}$$

Using the inequality (8.32) in (8.31),

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

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

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

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

If (8.29) holds, then $\{d(t)\} \to 0$ must hold. We also have from (8.28) that

$$\{x(t+1) - x(t)\} \to 0, \tag{9.33}$$

and from (8.32) that

$$\{x^i(t) - x(t)\} \to 0. \tag{9.34}$$

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 2.1). Then, (8.33) and (8.34) imply that $\{x(\tau_k)\} \to x^\infty$ and $\{x^i(\tau_k)\} \to x^\infty$ holds. From (8.28) it finally holds that $y^\infty = x^\infty$, that is, $x^\infty \in \Omega$. The last statement follows as in Theorem 1. □

4.3 Qualitative Analysis

The upper bound

$$\overline{\ell} = \frac{2 \min_{i \in \mathcal{C}} \{m_{\Phi_i}\}}{M_{\nabla f}\,(1 + [|\mathcal{C}|+1]P)} \tag{9.35}$$

on the maximal allowed step length provides interesting information on the relationships between properties of the problem, as quantified by the constant

$M_{\nabla f}$, the maximal asynchronism allowed, P, the number of independent processors used (or, variable components), $|\mathcal{C}|$, and the resulting maximal choice of 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 (8.32) is zero; by tracing the proof, the resulting upper bound can be shown to equal that in the synchronized approach (cf. Theorem 10).

Keeping $\bar{\ell}$ fixed, (8.35) 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 $\bar{\ell}$ 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$_{\Phi_i}$] as a function of their size and on the speed-up obtained from an increased amount of parallelism.

Summarizing the results obtained for the three algorithm versions, we have from (8.27) and (8.29) that

$$\frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j}\}}{M_{\nabla f}\,(1+[|\mathcal{C}|+1]P)} < \frac{2\min_{j\in\mathcal{C}}\{m_{\Phi_j^t}\}}{M_{\nabla f}} \leq \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}, \qquad i \in \mathcal{C}. \tag{9.36}$$

The relation (8.36) quantifies the intuitive idea 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 can 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 better describe the effect of coupling between the independent components x_i in the objective function on the maximal allowed asynchronism.

Assume that there is a positive constant M with

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

If f is separable with respect to the partition of $\Re^n$ defined by (8.1), then $M = 0$; a small value of M then indicates that the problem is weakly coupled. (Note that $M \leq M_{\nabla f}$ holds.)

We also introduce, in Assumption 2, the condition that

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

This assumption states that processor i always uses the most recent value of its own component x_i of x, and is in [72] referred to as a computational *non-redundancy*. This condition holds in general when no variable component is updated simultaneously by more than one processor.

By using these new assumptions in the equality (8.30), $x_i(t) = x_i^i(t)$ 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 (8.31) may be replaced by the smaller constant M. The upper bound on the step length (assuming that $P > 1$) may then be shown to be

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

Thus, for a fixed step, the asynchronism should be inversely proportional to the degree of coupling between the components x_i 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, with an anticipated gain in efficiency.

The convergence analysis shows under which circumstances the three versions produce the correct output. We have tried to give the mildest conditions possible; the conditions are, however, sufficient only, and in some instances, milder assumptions may therefore be possible to obtain.

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 [72], computations are *redundant*, that is, more than one processor may update the same variable component simultaneously.

Whenever Rule R is valid using unit steps and the function f satisfies some additional properties, the sequential and synchronous parallel algorithms may be shown to be linearly convergent (see [54]); we conjecture that this is true for the partially asynchronous model also.

The results obtained in this paper may—for the most part—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 (8.1), that is, the objective 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 [52, Sec. 3.12]).

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

$$F(x^*)^{\mathrm{T}}(x - x^*) \geq 0, \qquad \forall x \in X, \tag{9.37}$$

where $F : X \mapsto \Re^n$ is a continuous mapping which is neither the gradient of a real-valued function f nor a mapping of the form $[\nabla_x L^{\mathrm{T}}, -\nabla_y L^{\mathrm{T}}]^{\mathrm{T}}$ for some saddle function $L : X \times Y \mapsto \Re$. The updating steps of the CA algorithms of this paper are then replaced by (approximate) line searches with respect to merit functions for (8.37); see [52, 37] for more details.

6 EXERCISES

(1) Consider the problem of finding a point in the intersection of the closed and convex sets $X_i \subset \Re^n$, $i = 1, 2, \ldots, m$. Reformulate this problem into the form [CDP].

(2) Consider the problem of minimizing the function $x \mapsto \sum_{i=1}^m f_i(x)$ over the intersection defined in **(1)**, where $f_i : \Re^n \mapsto \Re$ is a strictly convex C^1 function. Formulate this problem in the form [SC].

(3) Consider the problem of minimizing the function $(x, y) \mapsto f(x) + g(y)$ subject to the constraints $Ax \leq b(y)$, where $f : \Re^n \mapsto \Re$, $g : \Re^s \mapsto \Re$, and $b : \Re^s \mapsto \Re^p$, with f, g and $-b_i$, $i = 1, 2, \ldots, p$, being convex in C^1, and $A \in \Re^{p \times n}$ is block diagonal with $|\mathcal{C}|$ blocks. Propose at least two strategies for decomposing this problem into problems of the form [CDP].

(4) Let $f : X \mapsto \Re$ be in C^1 on X with a Lipschitz continuous gradient on X, and $\Phi : X \mapsto \Re^n$ be continuous and strongly monotone on X. Under which conditions on Φ will Rules A and R, respectively, yield unit steps?

(5) Let $r^t : X \mapsto \Re^n$ be a strongly monotone gradient mapping. Show that in the regularization algorithm,

$$f(x^t + d^t) - f(x^t) \leq -m_{r^t}/(2\gamma_t)\|d^t\|^2.$$

(6) Give the cost approximating mappings Φ^t that characterize the decomposition versions of the algorithms given in Table 1.

(7) Suggest a rule for choosing indices in an essentially cyclic CA algorithm for [TAP].

(8) Verify that any truncated CA algorithm defined by a continuous function $\hat{\varphi}_i$ of the form $\hat{\varphi}_i(y_i, x_i)$, convex and differentiable with respect to y_i, together with a closed updating step which yields descent, has an algorithmic map that satisfies Assumption 1 whenever X is bounded.

(9) Construct the algorithmic map A of the truncated essentially cyclic CA algorithm.

(10) Determine the largest value of γ possible in the cyclic gradient projection algorithm.

(11) Consider the *saddle point problem* of finding $(x^*, y^*) \in X \times Y$ such that

[SPP] $\qquad L(x^*, y) \leq L(x^*, y^*) \leq L(x, y^*), \qquad \forall (x, y) \in X \times Y,$

where $X \subseteq \Re^n$ and $Y \subseteq \Re^m$ are nonempty, closed and convex sets and $L : X \times Y$ is a differentiable convex-concave function on $X \times Y$. Let the function $\varphi : X \times Y$ be of the form $\varphi(x, y) = \varphi_x(x) - \varphi_y(y)$, where $\varphi_x : X \mapsto \Re$ and $\varphi_y : Y \mapsto \Re$ are convex functions in C^1 on X and Y, respectively. Letting $Z = X \times Y$ and $z = (x, y)$, at $z^k \in Z$, we introduce the cost approximated saddle function L_φ given by $L_\varphi(z) = \varphi(z) + L(z^k) - \varphi(z^k) + [\nabla L(z^k) - \nabla\varphi(z^k)]^{\mathrm{T}}(z - z^k)$. Show that the associated saddle problem, [SPP$_\varphi$], is equivalent to evaluating the function $\psi(z^k) = \sup_{v \in Y} L_\varphi(x^k, v) - \inf_{w \in X} L_\varphi(w, y^k)$. Further, establish that the function $\psi : Z \mapsto \Re$ is a *merit function* (or, *gap function*) for [SPP], that is, that $\psi(z) \geq 0$ for all $z \in Z$ and $\psi(z) = 0$ if and only if z solves [SPP].

(12) Consider the saddle point problem [SPP] above, and let L be extended linear-quadratic, that is, $L(x, y) = \frac{1}{2}x^{\mathrm{T}}Px + p^{\mathrm{T}}x - \frac{1}{2}y^{\mathrm{T}}Qy - q^{\mathrm{T}}y + y^{\mathrm{T}}Rx$, where P and Q are symmetric and positive definite matrices. Also, let $\varphi_x(x) = \frac{1}{2}x^{\mathrm{T}}Ax$ and $\varphi_y(y) = \frac{1}{2}y^{\mathrm{T}}By$ for some symmetric and positive definite matrices A and B. Show that for any $z \in Z$ not solving [SPP], the solution of [SPP$_\varphi$] yields a direction of descent for ψ. Construct a synchronized parallel CA algorithm for [SPP] as well as a sequential version. Note the features of the Armijo step length rule.

Acknowledgements

This research was partly supported by a postdoctoral grant from the Swedish Research Council for Engineering Sciences (TFR) (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] G. M. Baudet, *Asynchronous iterative methods for multiprocessors*, J. ACM, 25 (1978), pp. 226–244.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, New York, NY, second ed., 1993.

[4] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, San Diego, CA, 1982.

[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] 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.

[22] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Programming, 55 (1992), pp. 293–318.

[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 non-separable 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] M. R. HESTENES, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.

[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] L. S. LASDON, *Optimization Theory for Large Systems*, Macmillan, New York, NY, 1970.

[39] F. A. LOOTSMA AND K. M. RAGSDELL, *State-of-the-art in parallel nonlinear optimization*, Parallel Comput., 6 (1988), pp. 133–155.

[40] D. G. LUENBERGER, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, second ed., 1984.

[41] 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.

[42] G. G. L. MEYER, *Asymptotic properties of sequences iteratively generated by point–to–set maps*, Math. Programming Study, 10 (1979), pp. 115–127.

[43] 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.

[44] 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.

[45] A. MIGDALAS, *Cyclic linearization vs. Frank–Wolfe decomposition for nonlinear problems over Cartesian product sets*, unpublished note, Department of Mathematics, Linköping Institute of Technology, Linköping, Sweden, 1990.

[46] A. NAGURNEY, *Network Economics: A Variational Inequality Approach*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[47] S. S. NIELSEN AND S. A. ZENIOS, *Massively parallel algorithms for singly constrained convex programs*, ORSA J. Comput., 4 (1992), pp. 166–181.

[48] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, 1970.

[49] 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.

[50] M. PATRIKSSON, *Partial linearization methods in nonlinear programming*, J. Optim. Theory Appl., 78 (1993), pp. 227–246.

[51] M. PATRIKSSON, *A unified description of iterative algorithms for traffic equilibria*, European J. Oper. Res., 71 (1993), pp. 154–176.

[52] M. PATRIKSSON, *A unified framework of descent algorithms for nonlinear programs and variational inequalities*, Doctoral dissertation, Department of Mathematics, Linköping Institute of Technology, Linköping, Sweden, 1993.

[53] M. PATRIKSSON, *Cost approximation: A unified framework of descent algorithms for nonlinear programs*, report, Department of Mathematics, Box 354350, University of Washington, Seattle, WA 98195-4350, 1994.

[54] M. PATRIKSSON, *Decomposition methods for differentiable optimization problems over Cartesian product sets*, Comput. Optim. Appl. (to appear).

[55] M. PATRIKSSON, *The Traffic Assignment Problem: Models and Methods*, VSP, Utrecht, 1994.

[56] E. R. PETERSEN, *A primal-dual traffic assignment algorithm*, Management Sci., 22 (1975), pp. 87–95.

[57] 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.

[58] M. J. D. POWELL, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, NY, 1969, pp. 283–298.

[59] R. T. ROCKAFELLAR, *Local boundedness of nonlinear, monotone operators*, Michigan Math. J., 16 (1969), pp. 397–407.

[60] R. T. ROCKAFELLAR, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[61] R. T. ROCKAFELLAR, *On the maximality of sums of nonlinear monotone operators*, Trans. Amer. Math. Soc., 149 (1970), pp. 75–88.

[62] R. T. ROCKAFELLAR, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.

[63] R. T. ROCKAFELLAR, *Network Flows and Monotropic Optimization*, John Wiley & Sons, New York, NY, 1984.

[64] S. TISHYADHIGAMA, E. POLAK, AND R. KLESSIG, *A comparative study of several general convergence conditions for algorithms modeled by point–to–set maps*, Math. Programming Study, 10 (1979), pp. 172–190.

[65] 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.

[66] P. TSENG, *Decomposition algorithm for convex differentiable minimization*, J. Optim. Theory Appl., 70 (1991), pp. 109–135.

[67] P. TSENG, *On the rate of convergence of a partially asynchronous gradient projection algorithm*, SIAM J. Optim., 1 (1992), pp. 603–619.

[68] 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.

[69] 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.

[70] J. N. TSITSIKLIS AND D. P. BERTSEKAS, *Distributed asynchronous optimal routing in data networks*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 325–332.

[71] A. ÜRESIN AND M. DUBOIS, *Sufficient conditions for the convergence of asynchronous iterations*, Parallel Comput., 10 (1989), pp. 83–92.

[72] 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.

[73] J. VAN TIEL, *Convex Analysis: An Introductory Text*, John Wiley & Sons, Chichester, U.K., 1984.

[74] N. Zadeh, *A note on the cyclic coordinate ascent method*, Management Sci., 16 (1969/1970), pp. 642–644.

[75] W. I. Zangwill, *Nonlinear Programming: A Unified Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[76] E. Zeidler, *Nonlinear Functional Analysis and its Applications II/B: Nonlinear Monotone Operators*, Springer-Verlag, New York, NY, 1990.

[77] S. Zenios and Y. Censor, *Massively parallel row-action algorithms for some nonlinear transportation problems*, SIAM J. Optim., 1 (1991), pp. 373–400.