Published in final edited form as:

ACM Trans Math Softw. 2010 April ; 37(2): 14–. doi:10.1145/1731022.1731024.

Efficient Controls for Finitely Convergent Sequential Algorithms

Wei Chen and

Department of Computer Science, Graduate Center, City University of New York, 365 Fifth Avenue,
New York, NY 10016, USA; wchen@gc.cuny.edu

Gabor T. Herman

Department of Computer Science, Graduate Center, City University of New York, 365 Fifth Avenue,
New York, NY 10016, USA; gabortherman@yahoo.com

Abstract

Finding a feasible point that satisfies a set of constraints is a common task in scientific computing: examples are the linear feasibility problem and the convex feasibility problem. Finitely convergent sequential algorithms can be used for solving such problems; an example of such an algorithm is ART3, which is defined in such a way that its control is cyclic in the sense that during its execution it repeatedly cycles through the given constraints. Previously we found a variant of ART3 whose control is no longer cyclic, but which is still finitely convergent and in practice it usually converges faster than ART3 does. In this paper we propose a general methodology for automatic transformation of finitely convergent sequential algorithms in such a way that (i) finite convergence is retained and (ii) the speed of convergence is improved. The first of these two properties is proven by mathematical theorems, the second is illustrated by applying the algorithms to a practical problem.

Categories and Subject Descriptors

G.1.0 [NUMERICAL ANALYSIS]: General—*Numerical algorithms*; G.1.3 [NUMERICAL ANALYSIS]: Numerical Linear Algebra—*Linear systems (direct and iterative methods); Sparse, structured, and very large systems (direct and iterative methods)*; G.4 [MATHEMATICAL SOFTWARE]: —*Algorithm design and analysis; Efficiency*

General Terms

Algorithms; Design; Experimentation; Performance; Theory

Additional Key Words and Phrases

Algebraic reconstruction technique; cyclic subgradient projections; finite convergence; feasibility problem; sequential algorithm

© 20YY ACM

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

1. INTRODUCTION

We use $\mathbb{Z}_{\geq}$ and $\mathbb{Z}_{>}$ to denote the set of nonnegative and of positive integers, respectively. For any $N \in \mathbb{Z}_{>}$, we use $\mathbb{R}^N$ to denote the N -dimensional Euclidean space.

The subject matter of this paper is *finitely convergent sequential algorithms*. Those who work in the field “know” what these are, but this is more by consensus than by precise definition. In the next paragraphs we present our definition of this concept from a computer science point of view.

First we specify the class of problems that these algorithms are supposed to solve. We deal with constraints in $\mathbb{R}^N$ of the form $g(x) \leq 0$, where $g : \mathbb{R}^N \rightarrow \mathbb{R}$. (Equivalently such constraints could have been specified using subsets of $\mathbb{R}^N$ or predicates on $\mathbb{R}^N$.) Let $\mathbb{G}$ be a set such that if $G \in \mathbb{G}$, then G is a nonempty finite set for which there exists an N in $\mathbb{Z}_{>}$ such that if $g \in G$, then $g : \mathbb{R}^N \rightarrow \mathbb{R}$. (We will use N_G to denote this N and use M_G to denote the number of elements in G .) We refer to such a $\mathbb{G}$ as a *problem set*. For a problem $G \in \mathbb{G}$, if a point $x \in \mathbb{R}^{N_G}$ satisfies $g(x) \leq 0$, for all $g \in G$, then x is said to be *feasible* for G . The tasks that are the subject matter of this paper are of the form: **Given** a $G \in \mathbb{G}$, **find** an $x \in \mathbb{R}^{N_G}$ that is feasible for G . Note that each problem set $\mathbb{G}$ requires its own algorithm.

A very simple example of such a problem set is the following.

$$\mathbb{G}_0 = \{\{\gamma_1, \gamma_2\} \mid \gamma_1, \gamma_2 : \mathbb{R} \rightarrow \mathbb{R}, \gamma_1(x) = -x, \gamma_2(x) = -x - 0.2\}. \quad (1)$$

This $\mathbb{G}_0$ has only one element and, for $G \in \mathbb{G}_0$, $M_G = 2$ and $N_G = 1$. The task associated with this example is to find a nonnegative real number. More complicated and useful examples are given in the next section. Those examples fall into the general class of convex feasibility problems; for a review see [Bauschke and Borwein 1996]. The methods discussed in that review are projection algorithms, as indeed are the algorithms of most of our examples in this paper. Such algorithms are widely used; for another example, see [Bauschke and Combettes 2003]. It should however be noted that the following definition of a sequential algorithm is general enough to include algorithms that would not be considered to be projection algorithms (an illustration is given below).

In what follows we often restrict our attention to *closed* problem sets, which have the following property: If $G \in \mathbb{G}$, $H \neq \emptyset$ and $H \subseteq G$, then $H \in \mathbb{G}$. (This is closure under taking nonempty subsets.) Note that $\mathbb{G}_0$ is not closed, since $\{\gamma_1\} \notin \mathbb{G}_0$.

In order to define our notion of a sequential algorithm on a problem set $\mathbb{G}$, we first introduce the more general notion of a *sequential scheme* on $\mathbb{G}$. It utilizes a set of *states* $\mathbb{S}$ that depends on the input $G \in \mathbb{G}$. In the description below $S^k \in \mathbb{S}$, for all $k \in \mathbb{Z}_{\geq}$.

Input:

$$G \in \mathbb{G}$$

Scheme:

1. $x^0 \leftarrow \sigma(Z_0)$
2. $S^0 \leftarrow \sigma(Z_1)$
3. $k \leftarrow 0$
4. Repeat
5. $S^k \leftarrow S^0$

6. Repeat
7. $g^k \leftarrow \sigma(Z_2(S^k, k, x^k))$
8. If $g^k(x^k) \leq 0$
9. then
10. $x^{k+1} \leftarrow x^k$
11. $S^{k+1} \leftarrow \sigma(Z_3(S^k, k, x^k, g^k))$
12. else
13. $x^{k+1} \leftarrow \sigma(Z_4(S^k, k, x^k, g^k))$
14. $S^{k+1} \leftarrow \sigma(Z_5(S^k, k, x^k, g^k))$
15. $k \leftarrow k + 1$
16. Until *Condition* (S^k) is true

In this description $\sigma(Z)$ denotes an arbitrary element from the nonempty set Z . (σ is called the *choice function* in nondeterministic algorithms.)

In order to turn the sequential scheme on $\mathbb{G}$ into a *sequential algorithm* on $\mathbb{G}$, we first must specify the choice of $\mathbb{S}$ for any $G \in \mathbb{G}$. After this we need to provide algorithmic specifications of Z_0 that has to be a nonempty subset of $\mathbb{R}^{NG}$, of Z_1 that has to be a nonempty subset of $\mathbb{S}$, and, for any $S \in \mathbb{S}$, $k \in \mathbb{Z}_{\geq}$ and $x \in \mathbb{R}^{NG}$, of $Z_2(S, k, x)$ that has to be a nonempty subset of G and, for any $g \in G$, of $Z_3(S, k, x, g)$ and of $Z_5(S, k, x, g)$ that have to be nonempty subsets of $\mathbb{S}$ and of $Z_4(S, k, x, g)$ that has to be a nonempty subset of $\mathbb{R}^{NG}$, and, for any $S \in \mathbb{S}$, of *Condition* (S) that takes the value *true* or *false* depending on S . An algorithm containing such nondeterministic assignment statements will have many *execution paths* (usually we use just *paths*, for short), each path consisting of a sequence of actions that may be taken as we execute the statements (deterministic or nondeterministic) of the algorithm.

As an example, for the $\mathbb{G}_0$ defined earlier, we let $\mathbb{S} = \{\emptyset\}$, $Z_0 = \mathbb{R}$, $Z_1 = Z_3(\emptyset, k, x, g) = Z_5(\emptyset, k, x, g) = \{\emptyset\}$, $Z_2(\emptyset, k, x) = \{\gamma_1, \gamma_2\}$, for any $k \in \mathbb{Z}_{\geq}$, $x \in \mathbb{R}$, $Z_4(\emptyset, k, x, \gamma_1) = \{x - 0.25\}$ and $Z_4(\emptyset, k, x, \gamma_2) = \{x + 0.75\}$, and *Condition*(S^k) = *false*. In the paths of this algorithm in which the constraints selected by $\sigma(\{\gamma_1, \gamma_2\})$ form the sequence $(\gamma_1, \gamma_2, \gamma_1, \gamma_2, \dots)$, the aim of the task will be achieved in a finite number of steps. On the other hand, in a path in which the constraints selected by $\sigma(\{\gamma_1, \gamma_2\})$ form the sequence $(\gamma_1, \gamma_1, \gamma_1, \gamma_2, \gamma_1, \gamma_1, \gamma_1, \gamma_2, \dots)$, a negative x^0 does not lead to a nonnegative x^k and the aim of the task is never achieved.

A further example, in which $\mathbb{S} \neq \{\emptyset\}$, is also an illustration of how the choice function enters into the algorithmic operation of a device, such as a computer that is connected to a network from which it receives a stream of input. The device of our simple illustration is the proposed new check-out counter of Nevada Supermarkets that, instead of giving change to customers, rounds up the purchase price to the nearest dollar, but gives a prize to a customer whenever the total amount of extra profits due to rounding up has reached $(10 + u)$ dollars, where u is an integer, $1 \leq u \leq U$. By choosing the u large enough and the prize cheap enough, Nevada Supermarkets will not only speed up the check-out process, but will also make extra profit. To model this in our framework, we set $\mathbb{S} = \{P, R\}$, where P signifies that either the previous customer was given a prize or we are dealing with the first customer at this check-out counter, while R signifies the alternative. We use $\mathbb{G} = \{G_1, \dots, G_U\}$ with, for $1 \leq u \leq U$, $G_u = G_{u,1} \cup G_{u,2}$ with, for $1 \leq v \leq 2$, $G_{u,v} = \{\gamma_{u,v,w} | 0 \leq w \leq 99\}$. The w signifies the amount (in cents) of the round up associated with a customer, it is to model the unpredictable variability in w that we have to resort to using a choice function in our algorithm. To complete its details, we define

$\gamma_{u,1,w}(x) = w + 1$, $\gamma_{u,2,w}(x) = 100(10 + u) - w - x$, $Z_0 = \{0\}$, $Z_1 = \{P\}$, $Z_2(S, k, x)$ is $G_{u,1}$ if $S = P$ and is $G_{u,2}$ otherwise, $Z_3(S, k, x, g) = \{P\}$, $Z_4(S, k, x, g)$ is $\{w\}$ if $g = \gamma_{u,1,w}$ and is $\{w + x\}$ if $g = \gamma_{u,2,w}$, $Z_5 = \{R\}$, and $Condition(S)$ is *true* if, and only if, $S = P$. It is easy to see that in the running of this algorithm, k signifies the customer number (starting with 1) and that customer should be given a prize if $Condition(S^k)$ is *true*. It can also be seen that, for all k , $0 \leq x^k < 100(10 + u)$ and, consequently, the constraint $\gamma_{u,2,0}(x^k) \leq 0$ is never satisfied. Thus this example is very different from the ones in the rest of this paper (which are the ones that motivated our research); in all those examples the aim is the satisfaction of all the constraints.

Let us call the execution paths of a sequential algorithm when provided with input G its G -paths. (Often, when G is understood, we just say path instead of G -path.) Each such G -path P produces, in particular, an infinite *control sequence* $c(P) = (g^0, g^1, g^2, \dots)$ of elements of G (that we also denote by $(g^k)_{k \in \mathbb{Z}_{\geq 0}}$), an infinite *solution sequence* $s(P) = (x^0, x^1, x^2, \dots)$ of elements of $\mathbb{R}^{NG}$ (that we also denote by $(x^k)_{k \in \mathbb{Z}_{\geq 0}}$), and an infinite *state sequence* $d(P) = (S^0, S^1, S^2, \dots)$ of elements of $\mathbb{S}$ (that we also denote by $(S^k)_{k \in \mathbb{Z}_{\geq 0}}$). We say that the control sequence is

1. *cyclic* if, for every $k \in \mathbb{Z}_{\geq 0}$, $\{g^k, g^{k+1}, \dots, g^{k+M_G-1}\} = G$;
2. *almost cyclic* if there exists a $C \in \mathbb{Z}_{> 0}$ such that, for every $k \in \mathbb{Z}_{\geq 0}$, $\{g^k, g^{k+1}, \dots, g^{k+C-1}\} = G$;
3. *repetitive* if, for every $k \in \mathbb{Z}_{\geq 0}$ and $g \in G$, there exists a $k' > k$, such that $g^{k'} = g$.

Note that a cyclic control sequence is almost cyclic with $C = M_G$, and an almost cyclic control sequence is repetitive. In our first example the control sequence $(\gamma_1, \gamma_2, \gamma_1, \gamma_2, \dots)$ is cyclic and $(\gamma_1, \gamma_1, \gamma_1, \gamma_2, \gamma_1, \gamma_1, \gamma_1, \gamma_2, \dots)$ is almost cyclic (and is also repetitive).

In the next three paragraphs, we provide definitions of the following concepts:

- finite convergence of a solution sequence,
- finite convergence of a sequential algorithm,
- finite convergence of a sequential algorithm under cyclic (respectively, almost cyclic or repetitive) control,
- the equivalence of two paths,
- the generalization of a sequential algorithm,
- the equivalence of two sequential algorithms.

We say that the solution sequence $(x^k)_{k \in \mathbb{Z}_{\geq 0}}$ is *finitely convergent* if, for some $K \in \mathbb{Z}_{\geq 0}$, $x^k = x^K$, for all $k \geq K$. Under these circumstances, we say that $(x^k)_{k \in \mathbb{Z}_{\geq 0}}$ *converges to* x^K . In our first example, if the control sequence is $(\gamma_1, \gamma_2, \gamma_1, \gamma_2, \dots)$, then the solution sequence is finitely convergent to a nonnegative real number; if the control sequence is $(\gamma_1, \gamma_1, \gamma_1, \gamma_2, \gamma_1, \gamma_1, \gamma_1, \gamma_2, \dots)$, then the solution sequence is not finitely convergent, unless $x^0 \geq 0$. In our second example, the solution sequence is finitely convergent only if, after some point, the purchase of every customer that comes to the check-out counter costs exactly some number of dollars (and hence there is no round up).

We say that a sequential algorithm on $\mathbb{G}$ is *finitely convergent* if, for every $G \in \mathbb{G}$, the solution sequence of every G -path is finitely convergent to a point feasible for G . We say that a sequential algorithm on $\mathbb{G}$ is *finitely convergent under cyclic* (respectively, *almost cyclic* or *repetitive*) *control* if, for every $G \in \mathbb{G}$ and G -path whose control sequence is cyclic (respectively, almost cyclic or repetitive) the solution sequence is finitely convergent to a point feasible for G . Note that the sequential algorithm we defined on $\mathbb{G}_0$ is finitely convergent under

cyclic control, but it is not necessarily finitely convergent under almost cyclic control (for example, when the control sequence is $(\gamma_1, \gamma_1, \gamma_1, \gamma_2, \gamma_1, \gamma_1, \gamma_1, \gamma_2, \dots)$).

Let A and A' be two sequential algorithms for the task $\mathbb{G}$. For any $G \in \mathbb{G}$, we say that a G -path P of A and a G -path P' of A' are *equivalent* if they have the same control sequence and the same solution sequence. We say that A' is a *generalization* of A if, for every $G \in \mathbb{G}$ and every G -path of A , there exists a G -path of A' equivalent to it. We say that A and A' are *equivalent* if both of them is a generalization of the other.

The major concern of this paper is: Given a task of the kind specified at the beginning of this section for which there exists a finitely convergent sequential algorithm, how should we specify the set $\mathbb{S}$ of states, the sets $Z_0, Z_1, Z_2, Z_3, Z_4, Z_5$ and the *Condition* that determine the control sequence of the execution paths so that the resulting algorithm is more *efficient* than other such algorithms using alternative specifications. We do not attempt in this paper to give a precise definition of “the efficiency of an algorithm.” We use the phrase in the intuitive sense of something that reflects the computational cost of the algorithm reaching a feasible point for a given problem.

2. ILLUSTRATIVE EXAMPLES

Many published algorithms can be reformulated so that they become sequential algorithms as defined in the previous section. We give two examples from the literature.

2.1 ART3 and Its Generalization

The first example that we use to illustrate the definitions of the previous section is the algorithm ART3 [Herman 1975] designed to find a common point of given hyperslabs. ART3 is one of the large class of Algebraic Reconstruction Techniques (ART) that has been the subject of very active research over the past decades; for a recent example, see [Gordon and Gordon 2008]. To bring ART3 in line with the notation used in this paper, we define a particular problem set $\mathbb{G}_A$ that contains all G that satisfy:

1. All $g \in G$ are of the form

$$g(x) = \begin{cases} l_g - \langle a_g, x \rangle, & \text{if } \langle a_g, x \rangle < l_g, \\ \langle a_g, x \rangle - u_g, & \text{if } \langle a_g, x \rangle > u_g, \\ \max(l_g - \langle a_g, x \rangle, \langle a_g, x \rangle - u_g), & \text{otherwise,} \end{cases} \quad (2)$$

where $l_g, u_g \in \mathbb{R}$, $l_g \leq u_g$, $a_g \in \mathbb{R}^{NG}$ and $\|a_g\| \neq 0$. (Note that each such function gives rise to a hyperslab

$$Q_g = \{x \in \mathbb{R}^{N_g} \mid l_g \leq \langle a_g, x \rangle \leq u_g\}, \quad (3)$$

such that that $x \in Q_g$ if, and only if, $g(x) \leq 0$.)

2. The set of all $x \in \mathbb{R}^{NG}$ that are feasible for G (equivalently, the intersection of all Q_g for $g \in G$) is full dimensional, in the sense that it is not a subset of any $(N_G - 1)$ -dimensional hyperplane in $\mathbb{R}^{NG}$.

This $\mathbb{G}_A$ is closed.

The choices that are made to turn the sequential scheme into the GART3 algorithm (which is a generalization of the ART3 algorithm of [Herman 1975]) are the following.

$$\mathbb{S} = \{\emptyset\}.$$

$$Z_0 = \mathbb{R}^{NG}.$$

$$Z_1 = \{\emptyset\}.$$

$$Z_2(\emptyset, k, x) = G.$$

$$Z_3(\emptyset, k, x, g) = \{\emptyset\}.$$

$$Z_4(\emptyset, k, x, g) = \{p_g(x)\}, \text{ where}$$

$$p_g(x) = x - \begin{cases} \frac{\langle a_g, x \rangle - (u_g + l_g)/2}{\|a_g\|^2} a_g, & \text{if } \langle a_g, x \rangle < l_g - (u_g - l_g)/2, \\ 2 \frac{\langle a_g, x \rangle - l_g}{\|a_g\|^2} a_g, & \text{if } l_g - (u_g - l_g)/2 \leq \langle a_g, x \rangle < l_g, \\ 2 \frac{\langle a_g, x \rangle - u_g}{\|a_g\|^2} a_g, & \text{if } u_g < \langle a_g, x \rangle \leq u_g + (u_g - l_g)/2, \\ \frac{\langle a_g, x \rangle - (u_g + l_g)/2}{\|a_g\|^2} a_g, & \text{if } u_g + (u_g - l_g)/2 < \langle a_g, x \rangle. \end{cases} \quad (4)$$

$$Z_5(\emptyset, k, x, g) = \{\emptyset\}.$$

$$\text{Condition}(\emptyset) = \text{false}.$$

The function p_g of (4) is illustrated in Figure 1.

As a consequence of the definition of $Z_2(\emptyset, k, x)$, given any $G \in \mathbb{G}_A$, every element of $\{(g^k)_{k \in \mathbb{Z}_+} \mid g^k \in G, \text{ for } k \in \mathbb{Z}_+\}$ is a control sequence of a G -path of GART3. The way ART3 is specified in [Herman 1975] restricts the G -paths to those whose control sequence is cyclic. A way of achieving this using our definition of a sequential algorithm is the following (this way of describing things has been chosen since it introduces a notation that makes easier in the next section the discussion of how an intuitively more efficient version can be automatically produced for a finitely convergent sequential algorithm.)

Let

$$\mathbb{F} = \{(h_1, h_2, \dots, h_U) \mid U \in \mathbb{Z}_+ \text{ and , for } 1 \leq u \leq U, h_u \in G\}. \quad (5)$$

$\mathbb{F}$ is the set of all finite sequences of elements of G including the empty sequence. Then we define ART3 as a sequential algorithm on $\mathbb{G}_A$ by making the choices (for $G \in \mathbb{G}_A$):

$$\mathbb{S} = \mathbb{F}.$$

$$Z_0 = \mathbb{R}^{NG}.$$

$$Z_1 = \{(g_1, g_2, \dots, g_{M_G}) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\}.$$

$$Z_2((h_1, h_2, \dots, h_U), k, x) = \{h_1\}.$$

$$Z_3((h_1, h_2, \dots, h_U), k, x, g) = \{(h_2, \dots, h_U, h_1)\}.$$

$$Z_4((h_1, h_2, \dots, h_U), k, x, g) = \{p_g(x)\}.$$

$$Z_5((h_1, h_2, \dots, h_U), k, x, g) = \{(h_2, \dots, h_U, h_1)\}.$$

$$\text{Condition}(S) = \text{false}, \text{ for all } S \in \mathbb{S}.$$

The consequence is that, given the same input, for every path of ART3, there exists a path of GART3 (whose control sequence is cyclic) equivalent to it. Therefore GART3 is a generalization of ART3. ART3 was proved to be a finitely convergent sequential algorithm on

$\mathbb{G}_A$ in [Herman 1975]. Hence it is also the case that GART3 is a finitely convergent sequential algorithm on $\mathbb{G}_A$ under cyclic control.

2.2 MCSP and Its Generalization

The second illustrative example is the Modified Cyclic Subgradient Projection (MCSP) method of [De Pierro and Iusem 1988]. For this, $\mathbb{G}_B$ is the set of all G , in which every element $g : \mathbb{R}^{NG} \rightarrow \mathbb{R}$ is a *convex function* (i.e., for all $x, y \in \mathbb{R}^{NG}$ and $\alpha \in [0, 1]$, $g(\alpha x + (1 - \alpha)y) \leq \alpha g(x) + (1 - \alpha)g(y)$) and there exists a positive number $\hat{\varepsilon}$ such that

$$g(x) + \hat{\varepsilon} \leq 0, \text{ for all } g \in G, \quad (6)$$

has a solution $x \in \mathbb{R}^{NG}$. Such a G is said to satisfy the *Slater condition*. Note that $\mathbb{G}_A \subset \mathbb{G}_B$ since any function $g \in G \in \mathbb{G}_A$ is a convex function, and full-dimensionality of the intersection implies that a ball with some positive radius $\hat{\varepsilon}$ can be fit into the intersection of hyperslabs determined by elements of G such that the center of the ball satisfies (6). The problem set $\mathbb{G}_B$ is also closed. The task of MCSP is to find, for a given $G \in \mathbb{G}_B$, an $x \in \mathbb{R}^{NG}$ that is feasible for G . This is called the *convex feasibility problem*.

A *subgradient* of a convex function $g : \mathbb{R}^N \rightarrow \mathbb{R}$ at the point $x \in \mathbb{R}^N$, is a vector $t \in \mathbb{R}^N$, that satisfies

$$\langle t, y - x \rangle \leq g(y) - g(x), \quad (7)$$

for all $y \in \mathbb{R}^N$. The set of all such subgradients is denoted by $\partial g(x)$. If g is differentiable at $x \in \mathbb{R}^N$, then its *gradient* $\nabla g(x)$ is the unique subgradient of g at x .

Let

$$\mathbb{E} = \left\{ \left((\alpha_k)_{k \in \mathbb{Z}_+}, (\varepsilon_k)_{k \in \mathbb{Z}_+} \right) \mid (\alpha_k)_{k \in \mathbb{Z}_+} \text{ is a sequence of } \alpha_k \in \mathbb{R} \text{ such that } 0 < \liminf \alpha_k \leq \limsup \alpha_k < 2, \right. \\ \left. (\varepsilon_k)_{k \in \mathbb{Z}_+} \text{ is a monotonically decreasing sequence of positive } \varepsilon_k \in \mathbb{R} \text{ such that } \varepsilon_k \rightarrow 0 \text{ as } k \rightarrow \infty \text{ and } \sum_{k=0}^{\infty} \varepsilon_k = \infty \right\}. \quad (8)$$

We turn the sequential scheme into the generalized MCSP algorithm (GMCSP) by making the following choices.

$$\mathbb{S} = \mathbb{E}.$$

$$Z_0 = \mathbb{R}^{NG}.$$

$$Z_1 = \mathbb{E}.$$

$$Z_2 \left(\left((\alpha_k)_{k \in \mathbb{Z}_+}, (\varepsilon_k)_{k \in \mathbb{Z}_+} \right), k, x \right) = G.$$

$$Z_3 \left(\left((\alpha_k)_{k \in \mathbb{Z}_+}, (\varepsilon_k)_{k \in \mathbb{Z}_+} \right), k, x, g \right) = \left\{ \left((\alpha_k)_{k \in \mathbb{Z}_+}, (\varepsilon_k)_{k \in \mathbb{Z}_+} \right) \right\}.$$

$$Z_4 \left(((\alpha_k)_{k \in \mathbb{Z}_\geq}, (\varepsilon_k)_{k \in \mathbb{Z}_\geq}), k, x, g \right) = \left\{ x - \alpha_k \frac{g(x) + \varepsilon_k}{\|t\|^2} t \mid t \in \partial_g(x) \right\}.$$

$$Z_5 \left(((\alpha_k)_{k \in \mathbb{Z}_\geq}, (\varepsilon_k)_{k \in \mathbb{Z}_\geq}), k, x, g \right) = \left\{ ((\alpha_k)_{k \in \mathbb{Z}_\geq}, (\varepsilon_k)_{k \in \mathbb{Z}_\geq}) \right\}.$$

$$\text{Condition } \left(((\alpha_k)_{k \in \mathbb{Z}_\geq}, (\varepsilon_k)_{k \in \mathbb{Z}_\geq}) \right) = \text{false}.$$

It easily follows from the results in [De Pierro and Iusem 1988] that the GMCS algorithm is a finitely convergent sequential algorithm on $\mathbb{G}_B$ under almost cyclic control.

3. A BASIC IDEA FOR IMPROVING EFFICIENCY

Consider the lines

$$(8) \quad \text{If } g^k(x^k) \leq 0$$

$$(9) \quad \text{then}$$

$$(10) \quad x^{k+1} \leftarrow x^k$$

$$(11) \quad S^{k+1} \leftarrow \sigma(Z_3(S^k, k, x^k, g^k))$$

in the sequential scheme of Section 1. Since the vector x^k does not change as a result of these steps, they are essentially just wasting computer time. For example, in ART3 (and similarly for any other finitely convergent sequential algorithm in which every path P is forced to have a cyclic control sequence), as the k approaches K , it is the case most of the time that $g^k(x^k) \leq 0$. Nevertheless, the control sequence $c(P)$ keeps cycling through all the M_G elements of G just to find out that $l_{g^k} \leq \langle a_{g^k}, x^k \rangle \leq u_{g^k}$ is satisfied. Since in practice both M_G and the dimension N_G of the space in which the inner product is taken are large, this can result in a considerable computational burden with very little to show for it. For this reason it appears to be a good idea to rewrite the algorithm in such a way that it is rarely the case that $g^k(x^k) \leq 0$.

Such considerations led us to introduce in [Herman and Chen 2008] the algorithm ART3+, which is a more efficient special case of GART3 than ART3 is. In the following sections we report on this algorithm in the more general context of this paper. It is inherent in the discussion above that if ART3 is to be made more efficient in the manner described, then the resulting control sequences will no longer be necessarily cyclic. The following theorem is proved based on an approach presented in our paper [Herman and Chen 2008] and will be used in the next section.

THEOREM 3.1. *The GART3 algorithm is finitely convergent for $\mathbb{G}_A$ under repetitive control.*

PROOF. Let $G \in \mathbb{G}_A$. In this proof we abbreviate M_G by M and N_G by N . We need to show that, for any G -path whose control sequence is repetitive, the solution sequence is finitely convergent to a point that is feasible for G .

Let $c(P) = (g^k)_{k \in \mathbb{Z}_\geq}$ be a repetitive control sequence for a G -path P , and let $s(P) = (x^k)_{k \in \mathbb{Z}_\geq}$. By the definition of $\mathbb{G}_A$, we know that $Q = \bigcap_{g \in G} Q_g$ is full dimensional.

We first observe that, for all $k \in \mathbb{Z}_\geq$,

$$x^{k+1} \in Q_{g^k} \tag{9}$$

and, for any $z \in Q \subseteq Q_{g^k}$,

$$\|x^{k+1} - z\| \leq \|x^k - z\|. \quad (10)$$

This is trivially so if $g^k(x^k) \leq 0$, and it is easily checked otherwise based on simple geometrical considerations on the nature of hyperslabs and the definitions in (4), see Figure 1. Hence the solution sequence $(x^k)_{k \in \mathbb{Z}_{\geq}}$ is bounded and so it has accumulation points. But it cannot have two different accumulation points x and y , since in that case we would have from (10) that, for all $z \in Q$, $\|x - z\| = \|y - z\|$ and so Q would be a subset of a hyperplane separating x and y , contradicting that it is full dimensional. It follows therefore that the solution sequence $(x^k)_{k \in \mathbb{Z}_{\geq}}$ is convergent to a point, let us call it x^* .

If $x^* \notin Q$, then $x^* \notin Q_g$ for at least one $g \in G$. Since Q_g is a closed convex set, there must exist a $K \in \mathbb{Z}_{\geq}$ such that $x^k \notin Q_g$, for all $k \geq K$. From the definition of repetitive control sequence, we see that we are bound to come across a $k \geq K$ such that $g^k = g$, and so (9) implies that $x^{k+1} \in Q_g$. This contradiction shows that $x^* \in Q$.

Now define the positive real numbers r_g , for all $g \in G$, by

$$r_g = \begin{cases} \min \{u_g - \langle a_g, x^* \rangle, \langle a_g, x^* \rangle - l_g\}, & \text{if } x^* \text{ is in the interior of } Q_g, \\ (u_g - l_g)/2, & \text{otherwise,} \end{cases} \quad (11)$$

and the ball B by

$$B = \left\{ x \in \mathbb{R}^N \mid \|x - x^*\| \leq \min_{g \in G} \frac{r_g}{\|a_g\|} \right\}. \quad (12)$$

By convergence, there is a $K \in \mathbb{Z}_{\geq}$ such that $x^k \in B$, for all $k \geq K$. We are now going to show that, for all $k \geq K$, $\|x^{k+1} - x^*\| = \|x^k - x^*\|$. This will complete our proof since it, combined with the convergence of $(x^k)_{k \in \mathbb{Z}_{\geq}}$ to x^* , implies that, for $k \geq K$, $x^k = x^* \in Q$.

So consider a $k \geq K$. We know that $x^k \in B$. If x^* is in the interior of Q_{g^k} , then (11) and (12) imply (consult Figure 1) that $x^k \in Q_{g^k}$, resulting in, $x^{k+1} = x^k$. The alternative is that x^* is not in the interior of Q_{g^k} , and so we must have that either $\langle a_{g^k}, x^* \rangle = l_{g^k}$ or $\langle a_{g^k}, x^* \rangle = u_{g^k}$. Consider the first case. By (11) and (12) we have either that $x^k \in Q_{g^k}$ and so $x^{k+1} = x^k$, or that $l_{g^k} - (u_{g^k}$

$- l_{g^k})/2 \leq \langle a_{g^k}, x^k \rangle < l_{g^k}$ and so $x^{k+1} = x^k - 2 \frac{\langle a_{g^k}, x^k \rangle - l_{g^k}}{\|a_{g^k}\|^2} a_{g^k}$, which combined with $\langle a_{g^k}, x^* \rangle = l_{g^k}$ implies that $\|x^{k+1} - x^*\| = \|x^k - x^*\|$ (again, consult Figure 1). The same result can be similarly derived if $\langle a_{g^k}, x^* \rangle = u_{g^k}$.

4. IMPROVING THE EFFICIENCY OF FINITELY CONVERGENT ALGORITHMS

As a generalization of the ART3 algorithm, we say that a sequential algorithm on a problem set $\mathbb{G}$ is *cyclic* if, given input $G \in \mathbb{G}$, the following are satisfied.

$\mathbb{S} = \mathbb{F} \times \mathbb{H}$, where $\mathbb{F}$ is defined in (5) and $\mathbb{H}$ is some auxiliary set of states.

$Z_0 = \mathbb{R}^{NG}$.

$Z_1 = \{(g_1, g_2, \dots, g_{M_G}) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\} \times Z_{\mathbb{H}}$, $Z_{\mathbb{H}} \subseteq \mathbb{H}$, $Z_{\mathbb{H}} \neq \emptyset$.

$Z_2(((h_1, h_2, \dots, h_U), H), k, x) = \{h_1\}$.

$$Z_3(((h_1, h_2, \dots, h_U), H), k, x, g) = \{((h_2, \dots, h_U, h_1), H)\}.$$

For any $H \in \mathbb{H}$,

$$Z_4((F_2, H), k, x, g) = Z_4((F_1, H), k, x, g), \quad (13)$$

for all $F_1, F_2 \in \mathbb{F}$, $k \in \mathbb{Z}_{\geq}$, $x \in \mathbb{R}^{NG}$, and $g \in G$.

For any $H_1 \in \mathbb{H}$ and $l \in \mathbb{Z}_{\geq}$, there exists $H_2 \in \mathbb{H}$ such that

$$Z_4((F_2, H_2), k, x, g) = Z_4((F_1, H_1), l+k, x, g), \quad (14)$$

for all $F_1, F_2 \in \mathbb{F}$, $k \in \mathbb{Z}_{\geq}$, $x \in \mathbb{R}^{NG}$, and $g \in G$.

$$Z_5(((h_1, h_2, \dots, h_U), H), k, x, g) = \{((h_2, \dots, h_U, h_1), H)\}.$$

$Condition(S) = false$, for all $S \in \mathbb{S}$.

It is easy to check that the control sequence of every path of a cyclic sequential algorithm is cyclic. To demonstrate the appropriateness of our definition of a cyclic sequential algorithm, we now give the specifications for two of them. The first we call CART3, it is essentially ART3 expressed in the form of the general definition. The other is CMCSPP, it is a cyclic version of MCSPP.

Given input $G \in \mathbb{G}_A$, the choices made for CART3 are:

$$\mathbb{S} = \mathbb{F} \times \{\emptyset\}.$$

$$Z_0 = \mathbb{R}^{NG}.$$

$$Z_1 = \{((g_1, g_2, \dots, g_{M_G}), \emptyset) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\}.$$

$$Z_2(((h_1, h_2, \dots, h_U), \emptyset), k, x) = \{h_1\}.$$

$$Z_3(((h_1, h_2, \dots, h_U), \emptyset), k, x, g) = \{((h_2, \dots, h_U, h_1), \emptyset)\}.$$

$$Z_4(((h_1, h_2, \dots, h_U), \emptyset), k, x, g) = \{p_g(x)\}.$$

$$Z_5(((h_1, h_2, \dots, h_U), \emptyset), k, x, g) = \{((h_2, \dots, h_U, h_1), \emptyset)\}.$$

$Condition(S) = false$, for all $S \in \mathbb{S}$.

If we plug in these specifications into the sequential scheme, we get that the cyclic algorithm CART3, for input $G \in \mathbb{G}_A$, is the following.

1. $x^0 \leftarrow \sigma(\mathbb{R}^{NG})$
2. $S^0 \leftarrow \sigma(\{((g_1, g_2, \dots, g_{M_G}), \emptyset) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\})$
3. $k \leftarrow 0$
4. Repeat
 5. $S^k \leftarrow S^0$
 6. Repeat
 7. $g^k \leftarrow h_1$, if $S^k = ((h_1, h_2, \dots, h_{M_G}), \emptyset)$
 8. If $g^k(x^k) \leq 0$
 9. then

10. $x^{k+1} \leftarrow x^k$
11. $S^{k+1} \leftarrow ((h_2, \dots, h_{M_G}, h_1), \emptyset)$, if $S^k = ((h_1, h_2, \dots, h_{M_G}), \emptyset)$
12. else
13. $x^{k+1} \leftarrow p_{g_k}(x^k)$
14. $S^{k+1} \leftarrow ((h_2, \dots, h_{M_G}, h_1), \emptyset)$, if $S^k = ((h_1, h_2, \dots, h_{M_G}), \emptyset)$
15. $k \leftarrow k + 1$
16. Until *false* is true

It is easy to see that this algorithm is a cyclic sequential one with $\mathbb{H} = \{\emptyset\}$. In particular, (13) and (14) are satisfied, since Z_4 in CART3 depends only on x and g . Also, it is clearly the case that CART3 is equivalent to the ART3 in [Herman 1975] and GART3 is a generalization of CART3.

The cyclic MCSP (CMCSP) uses $\mathbb{E}$ defined in (8) as the set $\mathbb{H}$ of auxiliary states. Given input $G \in \mathbb{C}\mathbb{B}$, the choices made for CMCSP are:

$$\mathbb{S} = \mathbb{F} \times \mathbb{E}.$$

$$Z_0 = \mathbb{R}^{N_G}.$$

$$Z_1 = \{(g_1, g_2, \dots, g_{M_G}) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\} \times \mathbb{E}.$$

$$Z_2 \left(((h_1, h_2, \dots, h_U), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})), k, x \right) = \{h_1\}.$$

$$Z_3 \left(((h_1, h_2, \dots, h_U), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})), k, x, g \right) = \left\{ ((h_2, \dots, h_U, h_1), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})) \right\}.$$

$$Z_4 \left(((h_1, h_2, \dots, h_U), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})), k, x, g \right) = \left\{ x - \alpha_k \frac{g(x) + \varepsilon_k}{\|t\|^2} t \mid t \in \partial_g(x) \right\}.$$

$$Z_5 \left(((h_1, h_2, \dots, h_U), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})), k, x, g \right) = \left\{ ((h_2, \dots, h_U, h_1), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})) \right\}.$$

$$\text{Condition} \left(((h_1, h_2, \dots, h_U), ((\alpha_k)_{k \in \mathbb{Z}_Z}, (\varepsilon_k)_{k \in \mathbb{Z}_Z})) \right) = \text{false}.$$

It is easy to see that this algorithm is a cyclic sequential one with $\mathbb{H} = \mathbb{E}$. In particular, (13) and (14) are satisfied, since Z_4 is independent of $(h_1, h_2, \dots, h_U)$ and the tail of any element in $\mathbb{E}$ is also in $\mathbb{E}$. GMCSP is a generalization of CMCSP.

The method by which ART3+ is obtained from ART3 in our paper [Herman and Chen 2008] generalizes to obtaining a sequential algorithm A^+ from a cyclic sequential algorithm A . The specifications for A^+ are exactly the ones for A , with the exception of Z_3 and the *Condition*, which are replaced by

$$Z_3^+ \left(((h_1, h_2, \dots, h_U), H), k, x, g \right) = \{((h_2, \dots, h_U), H)\}.$$

$$\text{Condition}^+(S) = \begin{cases} \text{true}, & \text{if } S = ((\cdot), H), \text{ for some } H \in \mathbb{H}, \\ \text{false}, & \text{otherwise.} \end{cases}$$

To see the purpose of introducing such a change, consider the discussion at the beginning of Section 3. Suppose that at a time line 7 is executed in a path either of the cyclic algorithm A or of A^+ (recall the definitions of a sequential scheme and a sequential algorithm in Section 1) $S^k = ((h_1, h_2, \dots, h_U), H)$. Then, by the condition on Z_2 in a cyclic algorithm, execution of line

7 will result in $g^k = h_1$. Now observe what happens if the condition in line 8 is satisfied. In either algorithm we get $x^{k+1} \leftarrow x^k$, and then we get to line 11. Due to the fact that the Z_3^+ of A^+ is different from the Z_3 of A , the behavior of the two algorithms diverge from each other: After executing line 11, we have $S^{k+1} = ((h_2, \dots, h_U), H)$ in the path of A^+ and $S^{k+1} = ((h_2, \dots, h_U, h_1), H)$ in the path of A . This means that as opposed to the cyclic nature of the control sequence of A , in the following repeated executions by A^+ of lines 7–15, we will not be assigning h_1 to g^k ; avoiding the wasteful behavior of the cyclic algorithm pointed out in Section 3. However, this brings about the danger that even if A is finitely convergent to a feasible point for any input from $\mathbb{G}$, A^+ may not share this desirable property: just because we have that $h_1(x^k) \leq 0$, it does not mean that $h_1(x^{k'}) \leq 0$, for all $k' > k$. The aim of changing *Condition* to *Condition*⁺ is to retain convergence to a feasible point. The next theorems specify the circumstances under which this aim is achieved.

THEOREM 4.1. *Let $\mathbb{G}$ be a closed problem set. If a cyclic sequential algorithm A on $\mathbb{G}$ is finitely convergent, then the control sequence of every path of A^+ is repetitive.*

PROOF. Let $\mathbb{G}$ be a closed problem set and A be a cyclic sequential algorithm that is finitely convergent on $\mathbb{G}$. Assume that $G^+ \in \mathbb{G}$ is such that the control sequence $c(P^+)$ of a G^+ -path P^+ of A^+ is not repetitive. We show that this assumption leads to a contradiction. We use the notations $c(P^+) = (f^l)_{l \in \mathbb{Z}_+}$, $s(P^+) = (y^l)_{l \in \mathbb{Z}_+}$, and $d(P^+) = (T^l)_{l \in \mathbb{Z}_+}$. It follows from the assumption that after some point in P^+ , *Condition*⁺(T^l) is never assigned the value *true*; for otherwise line 5 would set T^l to be $((f_1, f_2, \dots, f_{M_{G^+}}), H)$, with $\{f_1, f_2, \dots, f_{M_{G^+}}\} = G^+$ and $H \in \mathbb{H}$, and the inner loop (lines 7–15) would be executed, which would result in all the constraints in G^+ occurring as the next M_{G^+} elements in the control sequence $c(P^+)$. Consequently, after some point in P^+ , it cannot be the case that $T^l = ((), H)$, for some $H \in \mathbb{H}$, and the inner loop (lines 7–15) will be repeatedly executed without ever exiting from it. Since the definition of Z_3^+ implies that the execution of line 11 removes the first element of $(h_1, h_2, \dots, h_U)$, this means that after some later point, line 11 is never executed and $f^l(y^l) \leq 0$ is never satisfied. Consider now the situation following an execution of line 7 after this point. Suppose that at that time $l = l^+$ and $T^{l^+} = ((h_1, h_2, \dots, h_{M^+}), H_1)$, with $M^+ \geq 1$ and $\{h_1, h_2, \dots, h_{M^+}\} \subseteq G^+$. From the condition on Z_2 in a cyclic algorithm, we get that $f^{l^+} = h_1$. Also, if we define G to be the set $\{h_1, h_2, \dots, h_{M^+}\}$, then $G \in \mathbb{G}$ (recall that $\mathbb{G}$ is closed). Note that $M_G = M^+$ and $N_G = N_{G^+}$.

We are going to show that there is a G -path P of the cyclic sequential algorithm A such that $c(P) = (g^k)_{k \in \mathbb{Z}_+}$, $s(P) = (x^k)_{k \in \mathbb{Z}_+}$, $d(P) = (S^k)_{k \in \mathbb{Z}_+}$ and, for all $k \in \mathbb{Z}_+$, $g^k = f^{l^++k}$, $x^k = y^{l^++k}$ and $F(S^k) = F(T^{l^++k})$, where $F(S)$ is the first component of $S = (F, H)$. We prove this by inductively constructing a G -path P that satisfies the claim for all $k \in \mathbb{Z}_+$.

The input to algorithm A is G . Since in algorithm A , $Z_0 = \mathbb{R}^{N_G}$, it is possible for line 1 to return $x^0 = y^{l^+}$. Also, since in the cyclic algorithm A , $Z_1 = \{(g_1, g_2, \dots, g_{M_G}) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\} \times \mathbb{Z}_H$, it is possible for line 2 to return $S^0 = ((h_1, h_2, \dots, h_{M_G}), H_2)$, where H_2 has the property that $Z_4((F_2, H_2), k, x, g) = Z_4((F_1, H_1), l^+ + k, x, g)$ for all $F_1, F_2 \in \mathbb{F}$, $k \in \mathbb{Z}_+$, $x \in \mathbb{R}^{N_G}$, and $g \in G$, see (14). Therefore at this moment, and even after the execution of line 5,

$F(S^0) = F(T^{l^+}) = (h_1, h_2, \dots, h_{M_G})$ and by the condition on Z_2 , we have that, after the execution of line 7, $g^0 = f^{l^+} = h_1$. This shows that the base of the induction is true.

Now assume that the induction hypothesis (namely that after the execution of line 7, $g^k = f^{l^++k}$, $x^k = y^{l^++k}$ and $F(S^k) = F(T^{l^++k})$) holds, for some $k \in \mathbb{Z}_+$, and carry on constructing P so that the induction hypothesis holds for $k + 1$ after the next execution of line 7. We make the observation that, since neither A nor A^+ ever changes the second component of the state, we

have that $S^k = (F^k, H_2)$ and $T^{l^++k} = (F^k, H_2)$, where $F^k = F(S^k) = F(T^{l^++k})$. Because $f^l(y^l) \leq 0$ is not satisfied if $l \geq l^+$ in the execution of A^+ , and $g^k = f^{l^++k}$, $x^k = y^{l^++k}$, it must be the case that $g^k(x^k) > 0$. Hence, the execution of A goes to line 13. Since we have $Z_4((F^k, H_2), k, x^k, g^k) = Z_4((F^k, H_1), l^+ + k, y^{l^++k}, f^{l^++k})$, and Z_4 is the same for the two algorithms, it is possible to make the choice in line 13 of A so that we get $x^{k+1} = y^{l^++k+1}$. In line 14, by the common definition of Z_5 , $F(S^k) = F(T^{l^++k})$ results in $F(S^{k+1}) = F(T^{l^++k+1})$. Since, in both A and A^+ , $Condition()$ returns false, the execution returns back to line 7. In line 7, by the definition of Z_2 in the definition of a cyclic sequential algorithm, $F(S^{k+1}) = F(T^{l^++k+1})$ results in $g^{k+1} = f^{l^++k+1}$. This completes our inductive construction and proof.

However, by the assumption that the cyclic sequential algorithm A on $\mathbb{G}$ is finitely convergent, we know that the solution sequence $(x^k)_{k \in \mathbb{Z}_\geq}$ of every G -path P of A is finitely convergent to a feasible point. It follows that there exists a $K \in \mathbb{Z}_\geq$ such that, for all $k \geq K$, $g^k(x^k) \leq 0$ is satisfied for all $g \in G$. This contradicts the fact that $f^{l^++k}(y^{l^++k}) \leq 0$ is never satisfied for $k \geq 0$.

We define the algorithm A^0 for $G = \{g_1, g_2, \dots, g_{M_G}\} \in \mathbb{G}$ for a given cyclic algorithm A by the following specifications:

$$S^0 = H.$$

$$Z_0^0 = \mathbb{R}^{N_G}.$$

$$Z_1^0 = Z_{\mathbb{H}} \text{ (this is the same } Z_{\mathbb{H}} \text{ that is used in } A).$$

$$Z_2^0(H, k, x) = \{g_1, g_2, \dots, g_{M_G}\}.$$

$$Z_3^0(H, k, x, g) = \{H\}.$$

$$Z_4^0(H, k, x, g) = Z_4((F, H), k, x, g), \text{ for any } F \in \mathbb{F}; \text{ see (13).}$$

$$Z_5^0(H, k, x, g) = \{H\}.$$

$$Condition^0(S) = false.$$

As a consequence of the definition of $Z_2^0(H, k, x)$, given any $G \in \mathbb{G}$, every element of $\{(g^k)_{k \in \mathbb{Z}_\geq} \mid g^k \in G, \text{ for } k \in \mathbb{Z}_\geq\}$ is a control sequence of a G -path of A^0 . Therefore, it is easy to see that A^0 is a generalization of A .

LEMMA 4.2. *If A is a cyclic sequential algorithm on a problem set $\mathbb{G}$, then sequential algorithm A^0 is a generalization of the sequential algorithm A^+ .*

PROOF. We need to prove that, for every $G \in \mathbb{G}$ and every G -path P^+ of A^+ , there exists a G -path P^0 of A^0 equivalent to it. We use the notation $c(P^+) = (f^l)_{l \in \mathbb{Z}_\geq}$, $s(P^+) = (y^l)_{l \in \mathbb{Z}_\geq}$, and $d(P^+) = (T^l)_{l \in \mathbb{Z}_\geq}$. We are going to show that there is a G -path P^0 of the sequential algorithm A^0 with $c(P^0) = (g^k)_{k \in \mathbb{Z}_\geq}$, $s(P^0) = (x^k)_{k \in \mathbb{Z}_\geq}$, $d(P^0) = (S^k)_{k \in \mathbb{Z}_\geq}$ such that, for all $k \in \mathbb{Z}_\geq$, $g^k = f^k$, $x^k = y^k$ and $S^k = H(T^k) = H(T^0)$, where $H(T)$ is the second component of $T = (F, H)$. We do this by inductively constructing a G -path P^0 that satisfies the claim for all $k \in \mathbb{Z}_\geq$. First, it is possible for A^0 to pick $x^0 = y^0$ in line 1 (since $Z_0^0 = \mathbb{R}^{N_G}$), $S^0 = H(T^0)$ in line 2 (since $H(T^0) \in Z_{\mathbb{H}}$) and $g^0 = f^0$ in line 7 (since $Z_2^0(H, k, x) = \{g_1, g_2, \dots, g_{M_G}\}$). Now we assume the induction hypothesis (namely that after the execution of line 7, $g^k = f^k$, $x^k = y^k$ and $S^k = H(T^k) = H(T^0)$) holds, for some $k \in \mathbb{Z}_\geq$, and carry on constructing P^0 so that the induction hypothesis holds for

$k + 1$ after the next execution of line 7. Whether or not the condition in line 8 is satisfied, line 10 and line 13 can return $x^{k+1} = y^{k+1}$. Line 11 and line 14 will return $S^{k+1} = H(T^{k+1}) = H(T^0)$ because $S^{k+1} = S^k$ and $H(T^{k+1}) = H(T^k)$. After that, whether or not A^+ goes back to line 4 or line 6, $S^{k+1} = S^0 = H(T^0) = H(T^{k+1})$. Also it is possible for A^0 to return $g^{k+1} = f^{k+1}$ in line 7. This completes our inductive construction and proof.

THEOREM 4.3. *Let $\mathbb{G}$ be a closed problem set and A be a cyclic sequential algorithm on $\mathbb{G}$ such that the sequential algorithm A^0 on $\mathbb{G}$ is finitely convergent under repetitive control, then A^+ on $\mathbb{G}$ is finitely convergent.*

PROOF. The sequential algorithm A^0 on $\mathbb{G}$ is finitely convergent under cyclic control, which is a special case of repetitive control. Therefore the cyclic sequential algorithm A on $\mathbb{G}$ is finitely convergent, since the control sequence of any G -path of a cyclic sequential algorithm is cyclic. By Theorem 4.1, the control sequence of any G -path of A^+ for $G \in \mathbb{G}$ is repetitive. By the assumption of this theorem that the sequential algorithm A^0 on $\mathbb{G}$ is finitely convergent under repetitive control and Lemma 4.2, the algorithm A^+ on $\mathbb{G}$ is finitely convergent.

Consider now the cyclic sequential algorithm CART3, which is equivalent to ART3 as defined in Subsection 2.1. The algorithm CART3⁰ is exactly the algorithm GART3 and the algorithm CART3⁺ is equivalent to the algorithm ART3⁺ of [Herman and Chen 2008]. By Theorem 3.1, GART3 is finitely convergent on $\mathbb{G}$ under repetitive control. Hence, it follows from Theorem 4.3 that CART3⁺ (and, equivalently, ART3⁺) is finitely convergent on $\mathbb{G}_A$. Thus the main mathematical result of [Herman and Chen 2008] is an immediate consequence of the general theory that is put forth in the current paper.

However, in those cases when the sequential algorithm A^0 on $\mathbb{G}$ is known to be finitely convergent only under almost cyclic control, we cannot conclude based on Theorem 4.3 the finite convergence of A^+ on $\mathbb{G}$. We now present an alternate procedure that produces, for any cyclic sequential algorithm A and any $i_0 \in \mathbb{Z}_{\geq}$, a sequential algorithm $A^{++}(i_0)$ for which we can prove a theorem analogous to Theorem 4.3, but with the less restrictive premise.

The procedure consists of replacing $\mathbb{S}$, Z_1 , Z_2 , Z_3 , Z_4 , Z_5 and $Condition(S)$ in algorithm A with:

$$\mathbb{S}^{++} = \mathbb{F} \times \mathbb{H} \times \mathbb{Z}_{\geq}.$$

$$Z_1^{++} = \{(g_1, g_2, \dots, g_{M_G}) \mid \{g_1, g_2, \dots, g_{M_G}\} = G\} \times \mathbb{Z}_{\mathbb{H}} \times \{0\}.$$

$$Z_2^{++}(((h_1, h_2, \dots, h_U), H, i), k, x) = \{h_1\}.$$

$$Z_3^{++}(((h_1, h_2, \dots, h_U), H, i), k, x, g) = \{((h_2, \dots, h_U), H, i+1)\}.$$

$$Z_4^{++}(((h_1, h_2, \dots, h_U), H, i), k, x, g) = Z_4(((h_1, h_2, \dots, h_U), H), k, x, g).$$

$$Z_5^{++}(((h_1, h_2, \dots, h_U), H, i), k, x, g) = \{((h_2, \dots, h_U, h_1), H, i+1)\}.$$

$$Condition^{++}(S) = \begin{cases} true, & \text{if } S \text{ is of the form } (\mathbf{f}, \emptyset, i) \text{ with either } \mathbf{f} = () \text{ or } i > i_0, \\ false, & \text{otherwise.} \end{cases}$$

If we plug in these specifications into the algorithm CART3 that has been specified earlier, we get that the algorithm CART3⁺⁺(i_0), for input $G \in \mathbb{G}_A$ is the following.

1. $x^0 \leftarrow \sigma(\mathbb{R}^{NG})$
2. $S^0 \leftarrow \sigma(\{(g_1, g_2, \dots, g_{M_G}), \emptyset, 0\} \mid \{g_1, g_2, \dots, g_{M_G}\} = G\})$
3. $k \leftarrow 0$
4. Repeat

5. $S^k \leftarrow S^0$
6. Repeat
7. $g^k \leftarrow h_1$, if $S^k = ((h_1, h_2, \dots, h_U), \emptyset, i)$
8. If $g^k(x^k) \leq 0$
9. then
10. $x^{k+1} \leftarrow x^k$
11. $S^{k+1} \leftarrow ((h_2, \dots, h_U), \emptyset, i+1)$, if $S^k = ((h_1, h_2, \dots, h_U), \emptyset, i)$
12. else
13. $x^{k+1} \leftarrow p_{g^k}(x^k)$
14. $S^{k+1} \leftarrow ((h_2, \dots, h_U, h_1), \emptyset, i+1)$, if $S^k = ((h_1, h_2, \dots, h_U), \emptyset, i)$
15. $k \leftarrow k+1$
16. Until S^k is of the form $(\mathbf{f}, \emptyset, i)$ with either $\mathbf{f} = ()$ or $i > i_0$

In this algorithm, i keeps count of the number of executions of line 7 in the inner loop (irrespective of the truth of $g^k(x^k) \leq 0$, either line 11 or line 14 increases i by 1).

THEOREM 4.4. *If $\mathbb{G}$ is a problem set, A is a cyclic sequential algorithm on $\mathbb{G}$, $G \in \mathbb{G}$, $i_0 \in \mathbb{Z}_>$ and $i_0 > M_G$, then the control sequence of every G -path of $A^{++}(i_0)$ is almost cyclic.*

PROOF. If line 5 executed in any G -path of $A^{++}(i_0)$, then there can be at most $i_0 + 1$ consecutive executions of the inner loop (lines 7–15) before $Condition^{++}(S)$ is satisfied. This has to be followed by an other execution of line 5, which sets S to $((g_1, g_2, \dots, g_{M_G}), H, 0)$, with $\{g_1, g_2, \dots, g_{M_G}\} = G$ and $H \in \mathbb{H}$. The following M_G executions of the inner loop result in all the constraints in G occurring as the next M_G elements in the control sequence (recall that $i_0 < M_G$). Therefore the control sequence $\{g^k\}_{k \in \mathbb{Z}_\geq}$ of the G -path of $A^{++}(i_0)$ is almost cyclic.

LEMMA 4.5. *If A is a cyclic sequential algorithm on a problem set $\mathbb{G}$ and $i_0 \in \mathbb{Z}_>$, then A^0 is a generalization of the sequential algorithm $A^{++}(i_0)$.*

PROOF. The proof is similar to the proof of Lemma 4.2.

THEOREM 4.6. *If $\mathbb{G}$ is a problem set, A is a cyclic sequential algorithm on $\mathbb{G}$ such that the sequential algorithm A^0 on $\mathbb{G}$ is finitely convergent under almost cyclic control, $G \in \mathbb{G}$, $i_0 \in \mathbb{Z}_>$ and $i_0 > M_G$, then the solution sequence of every G -path of $A^{++}(i_0)$ on $\mathbb{G}$ is finitely convergent.*

PROOF. Under the assumptions of the theorem, it follows from Theorem 4.4 that the control sequence of every G -path of $A^{++}(i_0)$ is almost cyclic. The result now follows, since by Lemma 4.5 a solution sequence of a G -path of $A^{++}(i_0)$ is also a solution sequence of a G -path of A^0 on $\mathbb{G}$, which is assumed to be finitely convergent under almost cyclic control.

Our Theorem 3.1 regarding $CART3^0 = GART3$ and the theorem in [De Pierro and Iusem 1988] that implies that $CMCSP^0 = GCMCSP$ on $\mathbb{G}_B$ is finitely convergent under almost cyclic control can now be seen to provide the following.

COROLLARY 4.7. *If $G \in \mathbb{G}_A$ and $i_0 > M_G$, then the solution sequence of every G -path of $CART3^{++}(i_0)$ on $\mathbb{G}$ is finitely convergent. If $G \in \mathbb{G}_B$ and $i_0 > M_G$, then the solution sequence of every G -path of $CMCSP^{++}(i_0)$ on $\mathbb{G}$ is finitely convergent.*

5. EXPERIMENTS

There are many applications of the linear interval feasibility problem. One of these is intensity-modulated radiation therapy (IMRT) planning. In [Herman and Chen 2008] we gave some anecdotal evidence for the claim that the efficiency of ART3+ is better than that of ART3 when applied to problems arising from IMRT.

In this section we consider an alternative application: the conversion of pixel images to blob images, i.e., the finding of the blob coefficients to fit an image represented by pixel coefficients. Furthermore, rather than just providing some anecdotal examples, we carry out a comprehensive study that assigns statistical significance to claims of superiority. We first introduce the problem formulation, then explain the experimental setup, and finally report on the comparison results of applying CART3 (which is equivalent to ART3), CART3⁺ (which is equivalent to ART3+ in [Herman and Chen 2008]) and CART3⁺⁺(i_0) to solving such problems. The experiments used the software package SNARK09 [Davidi et al. 2009].

5.1 From Pixel Images to Blob Images

A $J \times J$ digitized image is one whose value in the interior of any pixel of a J^2 -element grid is uniform. Sometimes alternative representations of an image are superior. For example, in image reconstruction from projections [Herman 2009], we use blob basis functions in some series expansion methods to reduce artifacts in the reconstruction. Such a reduction is due to the fact that blob basis functions are smoother than pixel basis functions [Lewitt 1990].

Assume that we have a set of M basis functions $\{q_1, \dots, q_M\}$, whose linear combinations give us adequate approximations to images L . An example of such an approach is the $J \times J$ digitization. In that case $M = J^2$. We number the pixels from 1 to J^2 , and define the J^2 basis functions by

$$d_j(r, \phi) = \begin{cases} 1, & \text{if } (r, \phi) \text{ is inside the } j\text{th pixel,} \\ 0, & \text{otherwise,} \end{cases} \quad (15)$$

where (r, ϕ) are the polar coordinates. Then the $J \times J$ digitization of an image L is the image $\hat{L}$ defined by

$$\hat{L}(r, \phi) = \sum_{j=1}^{J^2} y_j d_j(r, \phi), \quad (16)$$

where y_j is the average value of L inside the j th pixel.

An alternative is to use blob basis functions [Lewitt 1990]. Let

$$b_{a,\alpha,\delta}(r, \phi) = \begin{cases} C_{a,\alpha,\delta} \left(1 - \left(\frac{r}{a}\right)^2\right) I_2 \left(\alpha \sqrt{1 - \left(\frac{r}{a}\right)^2}\right), & \text{if } 0 \leq r \leq a, \\ 0, & \text{otherwise,} \end{cases} \quad (17)$$

where I_m is the modified Bessel function of the first kind of order m , a is the nonnegative radius of the blob, α is a nonnegative real number that controls the blob's taper (shape) and $C_{a,\alpha,\delta}$ is the multiplying constant:

$$C_{a,\alpha,\delta} = \frac{\sqrt{3}\delta^2\alpha}{4\pi a^2 I_3(\alpha)}. \quad (18)$$

It is easy to see that the function defined above is circularly symmetric since its value does not depend on ϕ , and it is a smooth function since its first derivatives are continuous everywhere. For the purpose of this discussion, the parameters a , α and δ , and hence the function $b_{a,\alpha,\delta}$, are fixed. Consider a hexagonal grid (with spacing determined by δ), and let N be the number of grid points that cover the image region. Then, we set the number M of basis function to N and, for $1 \leq n \leq N$, the blob basis function b_n is obtained from $b_{a,\alpha,\delta}$ by shifting it in the plane so that its center is moved from the origin to the grid point n , and a blob image is

$$\sum_{n=1}^N x_n b_n(r, \phi), \quad (19)$$

where the x_n is called the n th blob coefficient.

For some applications (e.g., [Herman 2009]), we sometimes need to convert a pixel image into a blob image. For this we need to solve J^2 linear interval inequalities:

$$y_j + \gamma_j \leq \sum_{n=1}^N x_n b_n(r_j, \phi_j) \leq y_j + \zeta_j, \text{ for } 1 \leq j \leq J^2, \quad (20)$$

where (r_j, ϕ_j) are the polar coordinates of the center of pixel j , y_j is the pixel value of pixel j , and $\gamma_j \leq 0$ and $\zeta_j \geq 0$ have some appropriately chosen values. For such a problem G , $M_G = J^2$, $N_G = N$, and $G \in \mathbb{G}_A$, provided that the solution space is full-dimensional. Therefore we can solve the problem G with CART3, CART3⁺ and CART3⁺⁺ (i_0), as long as $i_0 > M_G$.

5.2 The Brain Phantom

A phantom is put together by superimposing a number of elemental objects, placed at desired positions, at desired orientations and of desired size and density. The density of the image at any point is then defined as the sum of the densities associated with all the elemental objects within which the point lies.

The phantoms we used in the experiments are all simulations of a cross section of a human head that was reconstructed by CT [Herman 2009]. Based on this cross section we specified a skull enclosing the brain with ventricles, two small tumors, and a hematoma (blood clot) using five ellipses, eight segments of circles, and two triangles. We also simulated the occurrence of a textured large tumor and of inhomogeneity in our phantom. This was done by adding to the list of elemental objects a much longer list of additional elemental objects, each coinciding exactly with one pixel. To simulate random inhomogeneity, the densities assigned to these objects were randomly selected from a zero-mean Gaussian distribution. Also, SNARK09 enables us to add to the list of elemental objects pairs of objects with a specified probability P of assigning density d_1 to the left object and density d_2 to the right object. Thus we can randomly generate infinitely many similar, but not identical, brain phantoms.

We used SNARK09 [Davidi et al. 2009] to obtain the density in each of 243×243 pixels of size 0.0752 cm, for each randomly generated phantom. So $M_G = J^2 = 243 \times 243 = 59,049$. We used a hexagonal grid that resulted in blob images with $N_G = N = 51,151$.

In order to see clearly the features in the interior of the skull, we use zero (black) to represent the value 0.204 (or anything less) and 255 (white) to represent the value 0.21675 (or anything more). This way the small change in density by 0.001 corresponds to a change of twenty in display grayness, which is visible. The images we displayed below using this method are referred to as *windowed images*. Figure 2 shows one of the phantoms together with its windowed image and the blob images and their corresponding windowed images as provided for that phantom by CART3, CART3⁺ and CART3⁺⁺, respectively. Figure 3 shows the blob images as solved by CART3 and CART3⁺ for another one of the phantoms, and the plots of the 131th columns of the blob images solved by CART3 and CART3⁺, respectively, against that of that phantom.

5.3 Task-Oriented Comparisons of Algorithm Performance

The task was to convert brain phantoms into their blob representations. We did four experiments: two pair-wise comparisons between CART3 and CART3⁺ and between CART3 and CART3⁺⁺. For each pair, we ran twice with different order to avoid bias due to caching. In each experiment, using current time as seed for the random number generator, 30 phantoms were generated and were solved by both algorithms that were being compared. The timings were averaged for the 30 runs. SNARK09 also reported on the significance (as measured by the P-value) for rejecting the null hypothesis that the two algorithms are equally fast in favor of the alternative hypothesis that the one with faster average performance is in fact faster.

All of the three algorithms CART3, CART3⁺ and CART3⁺⁺ initialize x^0 to be the vector all of whose components are zero (see line 1 of the scheme) and all of them initialize S^0 to be an efficient ordering (in the sense advocated in [Herman and Meyer 1993]) of the constraints associated with the pixel indices (see line 2 of the scheme). The variant of CART3⁺⁺ that we used in our experiments is CART3⁺⁺(i_0) with $i_0 = M_G + 70,000 = 59,049 + 70,000 = 129,049$, see Section 4. Due to the careful choice of the tolerances γ_j and ζ_j , for $1 \leq j \leq J^2 = M_G$, all the algorithm runs converged to a feasible solution.

The experiments were conducted on an Intel Xeon 1.7GHz processor, 1G RAM workstation. Table I gives the average timings of 30 runs for both algorithms, the speedup ratios, and the P-values (which indicate that the null hypothesis of equal average performance can be rejected with extreme significance) for each experiment.

6. CONCLUSIONS

We proposed two automatic transformation procedures for converting a finitely convergent sequential algorithm into another sequential algorithm that (i) retains the property of finite convergence and (ii) is likely to be more efficient. As two examples, CART3⁺ and CART3⁺⁺(i_0) were obtained from ART3 for solving linear interval systems and CMCSPP⁺⁺(i_0) was obtained from MCSP for solving convex feasibility problems. From the performed experiments we conclude that in the application of converting pixel images to blob images both CART3⁺ and CART3⁺⁺(i_0) are statistically significantly faster than CART3 (which is equivalent to ART3), with a speedup ratio better than 1.4. These results support our claim that our proposed approach is appropriate for automated speeding up of finitely convergent sequential algorithms.

Acknowledgments

The work was supported by Award Number R01HL070472 from the National Heart, Lung, And Blood Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Heart, Lung, And Blood Institute or the National Institutes of Health.

We thank Yair Censor for many useful discussions on the topic of this paper.

REFERENCES

- Bauschke HH, Borwein JM. On projection algorithms for solving convex feasibility problems. *SIAM Review* 1996 Sep.;38(3):367–426.
- Bauschke HH, Combettes PL. Iterating Bregman retractions. *SIAM J. Optim* 2003;13(4):1159–1173.
- Davidi, R.; Herman, GT.; Klukowska, J. SNARK09: A Programming System for the Reconstruction of 2D Images from 1D Projections. New York: The CUNY Institute for Software Design and Development; 2009.
- De Pierro AR, Iusem AN. A finitely convergent row-action method for the convex feasibility problem. *Appl. Math. Optim* 1988 Jan.;17(1):225–235.
- Gordon D, Gordon R. CGMN revisited: Robust and efficient solution of stiff linear systems derived from elliptic partial differential equations. *ACM Trans. Math. Soft* 2008;35(3):1–27. (Article 18 Oct.).
- Herman GT. A relaxation method for reconstructing objects from noisy x-rays. *Math. Program* 1975 Dec.;8(1):1–19.
- Herman, GT. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. 2nd edition. Berlin: Springer; 2009.
- Herman GT, Chen W. A fast algorithm for solving a linear feasibility problem with application to intensity-modulated radiation therapy. *Linear Algebra Appl* 2008 Mar.;428(5–6):1207–1217. [PubMed: 18438465]
- Herman GT, Meyer LB. Algebraic reconstruction techniques can be made computationally efficient. *IEEE Trans. Med. Imaging* 1993 Sep.;12(3):600–609. [PubMed: 18218454]
- Lewitt RM. Multidimensional digital image representation using generalized Kaiser-Bessel window functions. *J. Opt. Soc. Am. A* 1990;7(10):1834–1846. [PubMed: 2231101]

Fig. 1.

The iterative step of GART3 from x^k to x^{k+1} , i.e., to $p_g(x^k)$.

Fig. 2.

Row 1 contains a phantom and its windowed image. Rows 2, 3 and 4 contain the blob images and their corresponding windowed images provided for the phantom by CART_3 , CART_3^+ and CART_3^{++} , respectively. The differences between the blob images produced by the three algorithms are so small that they cannot be identified based on such displays.

Fig. 3.

The first row contains windowed images of blob images provided by CART3 and by CART3⁺ for a phantom different from that of Figure 2. The second and third rows, respectively, show (dark) plots of the 131th columns of the blob images provided by CART3 and CART3⁺ against that of the phantom (light). The differences between the blob images produced by CART3 and CART3⁺ are so small that they cannot be identified based on such displays.

Table I

The average timings of the 30 runs.

	CART3	CART3 ⁺	CART3 ⁺⁺	Ratio	P-value
CART3 first	39.0 s	26.5 s		1.5	3.5×10^{-11}
CART3 ⁺ first	37.7 s	24.9 s		1.5	2.3×10^{-13}
CART3 first	39.5 s		24.6 s	1.6	1.4×10^{-13}
CART3 ⁺⁺ first	35.6 s		25.0 s	1.4	3.6×10^{-13}