SIAM J. OPTIM.
Vol. 14, No. 4, pp. 939–964

ON THE CONVERGENCE OF ASYNCHRONOUS PARALLEL PATTERN SEARCH*

TAMARA G. KOLDA† AND VIRGINIA J. TORCZON‡

Abstract. In this paper we prove global convergence for asynchronous parallel pattern search. In standard pattern search, decisions regarding the update of the iterate and the step-length control parameter are synchronized implicitly across all search directions. We lose this feature in asynchronous parallel pattern search since the search along each direction proceeds semiautonomously. By bounding the value of the step-length control parameter after any step that produces decrease along a single search direction, we can prove that all the processes share a common accumulation point and, if the function is continuously differentiable, that such a point is a stationary point of the standard nonlinear unconstrained optimization problem.

Key words. asynchronous parallel optimization, pattern search, unconstrained optimization, global convergence analysis

AMS subject classifications. 90C56, 90C30, 65K05, 65Y05, 68W15

DOI. 10.1137/S1052623401398107

1. Introduction.

Asynchronous parallel pattern search (APPS) was introduced in [5] as a way to solve in a parallel or distributed computing environment nonlinear optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x), \quad \text{where } f : \mathbb{R}^n \to \mathbb{R}. \tag{1.1}$$

In this paper, we prove that a subsequence of the sequence of iterates produced by APPS converges to a stationary point of (1.1), when f is continuously differentiable.

To do so, we build on the global convergence results for pattern search established in [7, 10, 11]. What distinguishes this analysis from the earlier work is the need to address the new concerns introduced by the asynchronism. The analyses in [7, 10, 11] rely on the fact that the more usual implementations of pattern search have complete knowledge of information acquired during the course of the search when making decisions about how to proceed. In contrast, APPS partitions out each search direction to a single process and, to eliminate idle time, does away with the close synchronization of the searches along each direction. This means that the search along the single direction governed by an individual process is allowed to proceed semiautonomously. By this we mean that each process is allowed to make its own

*Received by the editors November 13, 2001; accepted for publication (in revised form) September 8, 2003; published electronically May 25, 2004.

http://www.siam.org/journals/siopt/14-4/39810.html

†Computational Sciences and Mathematics Research Department, Sandia National Laboratories, Livermore, CA 94551–9217 (tgkolda@sandia.gov). The research of this author was sponsored by the Mathematical, Information, and Computational Sciences Division at the U.S. Department of Energy and by Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE-AC04-94AL85000.

‡Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187–8795 (va@cs.wm.edu). The research of this author was funded by the Computer Science Research Institute at Sandia National Laboratories and by the National Science Foundation under grant CCR-9734044.

decisions regarding the update of the iterate and the length of the next step, based only on the information currently available to it, even though that information may not be up-to-date with respect to the other processes. Further, there is no single controlling process. Instead, information between processes is exchanged intermittently so that eventually all processes learn of every reasonable candidate for the minimizer. The only assumption we make is that information about success (i.e., a decrease in the value of f) on one process reaches all other processes in a finite amount of time. We make no assumption about the order in which such information is received. Thus the processes act as a loose confederation of agents working toward a single goal: the identification of a stationary point of (1.1). The advantage of allowing processes to proceed semiautonomously is that we can eliminate synchronization barriers so that we achieve good computational performance when working in a parallel or distributed computing environment, as our tests in [5] demonstrate.

The critical issue for our analysis is that APPS makes decisions about updating the length of the next step and the best point in the absence of complete information about the progress of the searches along the other directions. Therefore, at any given time in the search, neither the value of the parameter each process uses to determine the length of the step nor the value of the best point may be the same across participating processes. Another minor aspect in which we differ from previous analysis is that we do not fix the contraction and expansion parameters used to update the step lengths. These differences require significant extensions to the analyses found in [7, 10, 11]. The key to safeguarding the overall outcome of the search lies in bounding the values which the parameter that controls the lengths of the steps is allowed to assume after any step that produces decrease on f (i.e., after a *successful* step).

In section 2 we describe a synchronous variant of parallel pattern search and use it to motivate APPS. In section 3 we outline APPS and introduce the extensive notation required for our analysis. We hasten to add that most of this bookkeeping, which is essential to our analysis, is not required in practice. A full treatment of the practical design and implementation of APPS is deferred to [5]. Since the notational overhead required for the analysis is significant, we refer interested readers to [6] for an example of APPS applied to a simple function, an illustration of the associated notation, and a discussion of those features of the asynchronous algorithms that most complicate the analysis. In this paper, we concentrate on the analysis, which is broken into four parts, covered in sections 5–8. In section 9 we close with some remarks regarding further extensions that could be made to the analysis.

Standard notation. We denote by $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$, and $\mathbb{N}$ the sets of real, rational, integer, and natural numbers, respectively.

We use $\text{pow}(\Lambda, \ell)$ to indicate that Λ is raised to the power ℓ, so that $\text{pow}(\Lambda, \ell) \equiv \Lambda^\ell$. We adopt this notational convention to eliminate any ambiguities that could arise when we introduce superscripts for use as indices.

2. Parallel pattern search. We start by considering a *synchronous* version of parallel pattern search (PPS) to clarify the notation and motivate APPS.

We assume that we have p independent processes, each of which is generating a sequence of trial points. We denote the set of processes as

$$\mathcal{P} = \{1, \ldots, p\}.$$

We work with a finite set of search directions

$$\mathcal{D} = \{d_1, \ldots, d_p\} = \{Bc_1, \ldots, Bc_p\}, \tag{2.1}$$

where

- $B \in \mathbb{R}^{n\times n}$ is a real nonsingular matrix,
- $c_i \in \mathbb{Q}^n$ for each $i \in \mathcal{P}$, and
- the vectors in the set $\mathcal{D}$ form a positive spanning set [7] for $\mathbb{R}^n$.

To each process $i \in \mathcal{P}$ we assign the constant search direction $d_i \in \mathcal{D}$. We constrain the vectors c_i, $i = \{1, \dots, p\}$, to the rationals to ensure that all iterates lie on a *rational lattice*, which, as we see in section 5, is required for the proof of Theorem 5.2. However, we allow a mapping of the rational vectors c_i to the real vectors d_i through the use of a fixed real nonsingular matrix B.

We denote by x_i^k the *best point* (i.e., one with the least function value) known by process i at iteration k. We denote by Δ_i^k the scalar that controls the length of the step taken along the direction d_i to construct a new trial point at iteration k. We refer to Δ_i^k as the *step-length control parameter*. For the synchronous version of pattern search, the subscript i on x and Δ is redundant since the synchronization ensures that the values of x_i^k and Δ_i^k are equivalent for all $i \in \mathcal{P}$; however, this subscript becomes meaningful in the asynchronous case, so we introduce the notation here for comparison.

Each process $i \in \mathcal{P}$ constructs a trial point by computing

$$x_i^k + \Delta_i^k d_i \tag{2.2}$$

and then evaluates f at this point. After the evaluation has finished on process i, process i broadcasts the result to all the other processes in $\mathcal{P}$ and then waits until it has received results from all the other processes in $\mathcal{P}$. This is the point of synchronization; no further action can be taken on process i until all the results from all the other processes in $\mathcal{P}$ are known. Once all p results are known to all p processes, a decision is made simultaneously as to which point is now best, and then x_i^k and Δ_i^k are updated to produce x_i^{k+1} and Δ_i^{k+1}. We assume that any ties are broken in a way that ensures all processes arrive at an identical choice for the new best point.

Because it is convenient for what follows, we replace the notion of iterations with the notion of occurrences at certain time steps as measured by a *global clock* like that used in other asynchronous convergence proofs; cf. [2]. Let the infinite set

$$\mathcal{T} = \{0, 1, 2, \dots\}$$

be the index of time steps. We assume that the time steps are of fine enough resolution that at most one *event* (i.e., a change in the best known point and/or the value of the step-length control parameter) occurs per time step, per process. In the synchronous case, iterations can be thought of as coarse time steps.

Using our global clock, we can represent the consequence of a single iteration, say k, for a single process, say $i \in \mathcal{P}$, on a timeline as illustrated in Figure 2.1. At time step t_0, process i starts a function evaluation at its trial point given by

$$x_i^{t_0} + \Delta_i^{t_0} d_i.$$

Observe that the notation introduced in (2.2) has changed. Now the time step replaces the iteration number in the superscript and, from now on, we use time steps as our indices. At time step t_1, process i finishes its evaluation of $f(x_i^{t_0} + \Delta_i^{t_0} d_i)$ and broadcasts its result to the remaining processes. We assume that at some time step t_2, all processes in $\mathcal{P}$ have received the results from all other processes, so each independently decides on the point that is now best. Since each process knows the

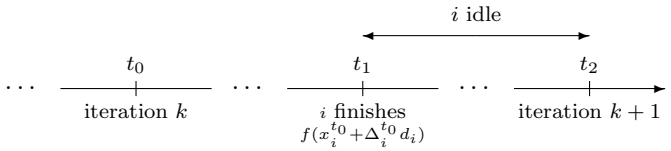

FIG. 2.1. *Timeline for synchronous pattern search for process i.*

results from all p processes in $\mathcal{P}$, and since ties are broken in a consistent fashion, all p processes will arrive at the same conclusion as to which point is now best. Each process then updates its copies of the best point and the step-length control parameter to obtain $x_i^{t_2}$ and $\Delta_i^{t_2}$. Iteration $k+1$ then begins. Note that from time step t_1 until time step t_2, process i is idle.

For process $j \in \mathcal{P}$, $j \neq i$, the procedure differs in only two respects. First, the trial point is calculated using a different search direction $d_j \in \mathcal{D}$ to yield

$$x_j^{t_0} + \Delta_j^{t_0} d_j.$$

Recall that $x_j^{t_0} = x_i^{t_0}$ and $\Delta_j^{t_0} = \Delta_i^{t_0}$ due to the synchronization. Second, we have no guarantee that the evaluation of f at the trial point will take the same number of time steps on process j as it did on process i. At one extreme is the possibility that the evaluation of f takes only a single time step, which would give us the scenario illustrated in Figure 2.2, where $\hat{t}_1$ denotes the time step at which the function evaluation on process j finishes. In this case, $\hat{t}_1 = t_0 + 1$ and process j is idle from time step $t_0 + 1$ to time step t_2. At the other extreme, we have the scenario in Figure 2.3, so that there is effectively no idle time on process j. Note that in this case we have assumed that the communication is instantaneous—our theory allows for this possibility as well as the possibility that communication may take up to a finite number of time steps.

We stress that even though the time required to finish a function evaluation may vary from process to process and from iteration to iteration, the synchronization ensures that, across all processes, iteration k begins at time step t_0 while iteration $k+1$ begins at time step t_2.

The goal of asynchronous parallel pattern search is to eliminate the synchronization since it potentially can waste CPU cycles, as our examples in Figures 2.1 and 2.2 demonstrate and our experimental evidence in [5] confirms. As we see in the next section, APPS allows each process to update its x_i^t and Δ_i^t independently whenever a function evaluation finishes and/or a new message arrives.

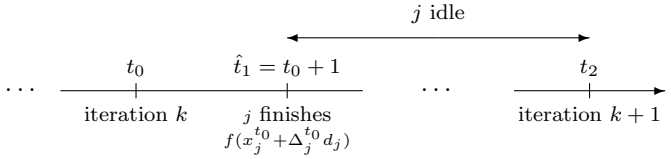

FIG. 2.2. *Timeline for synchronous pattern search for process j.*

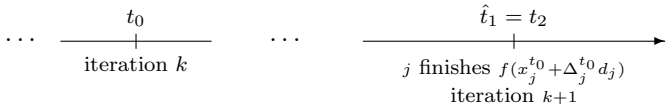

FIG. 2.3. *Alternate timeline for synchronous pattern search for process j.*

3. Asynchronous parallel pattern search. Like PPS, APPS [5] uses p processes collectively to solve (1.1). Each process is in charge of searching along a single search direction from its best known point, and the best known point and the value of the step-length control parameter are varied according to internal and external events. The difference is that individual processes in APPS no longer wait for information from the other processes before making a local decision as to the next best point. Once the decision is made, the process then updates its record of the best point and the step-length control parameter, constructs a new trial point, and immediately begins a new evaluation of the objective function.

Because we no longer have synchronization after every function evaluation, decisions now depend on the time step at which they are made. Therefore, we index according to the global clock described previously. We then define the following for each process $i \in \mathcal{P}$ and time step $t \in \mathcal{T}$:

$$x_i^t = \text{the best known point at time step } t \text{ for process } i, \text{ and}$$
$$\Delta_i^t = \text{the step-length control parameter at time step } t \text{ for process } i.$$

In APPS, the current values of the best point and the step-length control parameter can be different across processes at the same time step $t \in \mathcal{T}$. Therefore, the subscript i is no longer redundant, and it is possible that $x_i^t \neq x_j^t$ and/or $\Delta_i^t \neq \Delta_j^t$. On a single process $i \in \mathcal{P}$, we are guaranteed that at any time step $t \in \mathcal{T}$, $f(x_i^{t+1}) \leq f(x_i^t)$.

The values of x_i^t and Δ_i^t are not necessarily changed at every time step. Let

$$\mathcal{T}_i = \text{the set of time steps at which } x_i^t \text{ and/or } \Delta_i^t \text{ is changed,} \tag{3.1}$$

so that $\mathcal{T}_i \subseteq \mathcal{T}$. For each process $i \in \mathcal{P}$ we categorize each time step $t \in \mathcal{T}$ as either *successful* or *unsuccessful*. We also need to observe further distinctions within each of these two categories, which we detail in sections 3.2 and 3.3.

3.1. Assumptions. As a practical matter, we assume that at the start of the search the best point and the value of the step-length control parameter are equal for all $i \in \mathcal{P}$; that is, there exist $x^0 \in \mathbb{R}^n$ and $\Delta^0 \in \mathbb{R}$, $\Delta^0 > 0$ such that

$$x^0 = x_1^0 = x_2^0 = \cdots = x_p^0 \qquad \text{and} \qquad \Delta^0 = \Delta_1^0 = \Delta_2^0 = \cdots = \Delta_p^0. \tag{3.2}$$

We further assume that the value $f(x^0)$ is known by all processes.

As is standard for pattern search analysis, we assume

$$\mathcal{L}(x^0) = \{\, x \in \mathbb{R}^n \,:\, f(x) \leq f(x^0) \,\} \text{ is bounded.} \tag{3.3}$$

Also, to ensure that APPS always converges to a stationary point of (1.1), we assume that f is continuously differentiable on the closure of $\mathcal{L}(x^0)$, though we need this assumption only for our final result, Theorem 8.5.

We assume that $\mathcal{D}$, the set of search directions, is fixed and finite and of the form given in (2.1), with the conditions that B is a real nonsingular matrix, that $c_i \in \mathbb{Q}^n$ for each $i \in \mathcal{P}$, and that the vectors in the set $\mathcal{D}$ form a positive spanning set for $\mathbb{R}^n$.

We assume that the initial step-length control parameter is constrained by

$$0 < \Delta^{\min} \leq \Delta^0 \leq \Delta^{\max} < +\infty, \tag{3.4}$$

where $\Delta^{\min}$ and $\Delta^{\max}$ are constants. These same constants are used to bound Δ_i^t after any step that produces decrease on f (i.e., after any successful time step). This condition is given in (3.10) and is described fully in section 3.2.1.

We assume that both the maximum time for a function evaluation and the maximum time for a single communication are finite; we quantify those as

(3.5) $\eta =$ maximum number of time steps for evaluating f at a given x, and

(3.6) $\gamma =$ maximum number of time steps for communicating a message.

We assume that the minimum time for evaluation and communication are one and zero time steps, respectively.

3.2. Successful time steps. On process i, we characterize any time step $t \in \mathcal{T}$ at which we identify a point with a strictly lower value of f as *successful*. We further distinguish between *internal* and *external* successes depending on whether the information that identified improvement in the value of f was computed locally or received in the form of a message from another process; we detail these distinctions in sections 3.2.1 and 3.2.2.

We pay special attention to points that produce equal values of f since we must break ties in a consistent fashion. This becomes particularly critical in the asynchronous case since equivalent function values are likely to become known to each process at different time steps and perhaps in reverse order. To ensure the convergence of the overall search, we must ensure that when faced with equivalent function values, every one of the participating processes arrives at the same decision as to which of the points known to produce the same function value should be considered "best." Thus, we may have reason to classify some time steps as successful, even when they do not strictly improve the value of f. We describe such situations in more detail in section 3.2.2.

3.2.1. Internal successes. The first type of successful time step is an *internal success*, which can occur when a process finishes a function evaluation. Suppose that on process $i \in \mathcal{P}$ a function evaluation starts at some time step, say t_0, (using $x_i^{t_0}$ and $\Delta_i^{t_0}$ to generate the trial point) and finishes at some later time step, say t_1. We can represent this on a timeline as in Figure 3.1.

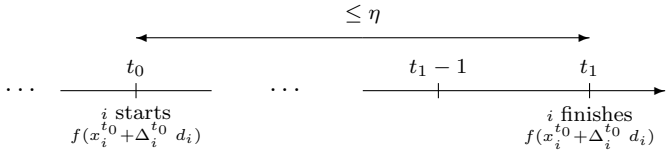

FIG. 3.1. *Timeline for asynchronous pattern search on process i.*

The time step t_1 is considered an internal success when the following condition is satisfied:

$$f\left(x_i^{t_0} + \Delta_i^{t_0}\, d_i\right) < f\left(x_i^{t_1-1}\right). \tag{3.7}$$

We compare $f(x_i^{t_0} + \Delta_i^{t_0}\, d_i)$ to $f(x_i^{t_1-1})$, rather than to $f(x_i^{t_0})$, since it is possible that $x_i^{t_1-1} \neq x_i^{t_0}$ due to an external success, which is described in the next section. When (3.7) is not satisfied, the time step is *unsuccessful*, as described in section 3.3. Otherwise, when (3.7) is satisfied, we say that time step $t_1 \in \mathcal{I}_i$, where

$$\mathcal{I}_i = \text{the set of internal successful time steps for process } i.$$

We then update x_i as follows:

$$x_i^{t_1} = x_i^{t_0} + \Delta_i^{t_0}\, d_i;$$

in other words, $x_i^{t_1}$ is set to the point that produced the best known function value. Further, we update the step-length control parameter Δ_i as follows:

$$\Delta_i^{t_1} = \lambda_i^{t_1} \Delta_i^{t_0},$$

where $\lambda_i^{t_1}$ is the *expansion parameter* for the update at time step t_1. Before we define the expansion parameter for the update, we first define the rational constant

$$\Lambda \in \mathbb{Q}, \quad \Lambda > 1, \tag{3.8}$$

which controls the scaling of all steps. Returning to the choice of λ_i^t, we require it to satisfy two conditions. The first condition is that λ_i^t be a nonnegative integer power of Λ; i.e.,

$$\lambda_i^t = \mathrm{pow}(\Lambda, k_i^t) \tag{3.9}$$

for some

$$k_i^t \in \{0, 1, 2, \dots\}.$$

Since $\Lambda > 1$ and k_i^t is nonnegative, $\lambda_i^t \geq 1$. The second condition on the choice of λ_i^t is that the new step-length control parameter must satisfy

$$0 < \Delta^{\min} \leq \Delta_i^t \leq \Delta^{\max} < +\infty, \tag{3.10}$$

where $\Delta^{\min}$ and $\Delta^{\max}$ are the same constants used in (3.4). Note that (3.10) applies *only* to updates associated with successful time steps. The bounds on Δ_i^t implicitly restrict the value of k_i^t that may be chosen in (3.9).

The lower bound on Δ is new to the asynchronous analysis; in section 4 we give an example that shows why this lower bound is necessary to ensure an accumulation point that is common to all processes. As for the upper bound on Δ, we could use the assumption that $\mathcal{L}(x^0)$ is bounded, given in (3.3), to yield an implicit upper bound on Δ, as is done in the analyses in [7, 10]. For convenience, here we assume the existence of an explicit upper bound and thus eliminate the dependence on f.

Once x_i and Δ_i are updated, process i broadcasts the new best point, its function value, and the new step-length control parameter to all the other processes in $\mathcal{P}$ for them to consider as a candidate for new best. Process i then proceeds with the construction and evaluation of $x_i^{t_1} + \Delta_i^{t_1} d_i$.

3.2.2. External successes. The other type of successful time step is an *external success*. Suppose that an internal success occurs on process i at time step t_1, as just described in section 3.2.1. Then at some time step $t_2 \geq t_1$, process j, $j \neq i$, receives the broadcast from process i with the new best point found by process i, along with its associated function value and step-length control parameter. We assume that process j can immediately assimilate the newly received information *even if it is currently in the midst of a function evaluation*. In the implementation described in [5], we achieve this by executing the function evaluation as a separate thread or process. We represent this example of an external success on the timeline in Figure 3.2.

There are three possibilities when process j receives a message from process i: the function value associated with the incoming point is either better than, equal to, or worse than the function value of the best point at the previous time step. Certainly, if $f(x_i^{t_1}) < f(x_j^{t_2-1})$ holds, then process j now has a new best point,

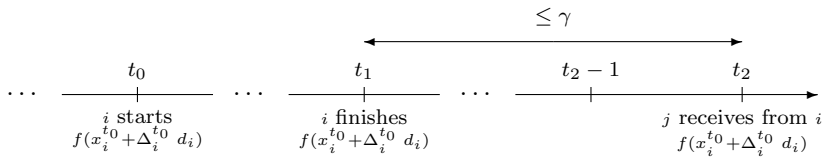

FIG. 3.2. *Timeline for asynchronous pattern search message from process i to process j.*

received from the external process i, and it should update its local values for the best point and the step-length control parameter in light of this new information. However, if $f(x_i^{t_1}) > f(x_j^{t_2-1})$, process j should simply discard the new information since $x_j^{t_2-1}$ is clearly better than $x_i^{t_1}$.

The interesting question is what to do when $f(x_i^{t_1}) = f(x_j^{t_2-1})$. To ensure the robustness of the search procedure, we define a comparison operator $\prec$. Given any $x, y, z \in \mathbb{R}^n$, $\prec$ denotes a comparison that satisfies the following two conditions:

1. $x \prec y$ and $y \prec z$ implies $x \prec z$, and
2. $x = y$ (i.e., neither $x \prec y$ nor $y \prec x$) only if $x[i] = y[i]$ for $i = 1, \ldots, n$, where the notation $x[\cdot]$ denotes the ith entry of the vector x.

We can use any definition for the comparison operator $\prec$ so long as it satisfies these two conditions. For example, we may use the following ordered elementwise comparison. We say $x \prec y$ if there exists $j \in \{1, \ldots, n\}$ such that $x[j] < y[j]$ and $x[i] = y[i]$ for $i = 1, \ldots, j-1$. Given a way to resolve ties, we are now ready to define an external success.

The time step t_2 is considered an external success if either

$$f\left(x_i^{t_1}\right) < f\left(x_j^{t_2-1}\right) \quad \text{or} \quad f\left(x_i^{t_1}\right) = f\left(x_j^{t_2-1}\right) \quad \text{and} \quad x_i^{t_1} \prec x_j^{t_2-1}. \tag{3.11}$$

If (3.11) is satisfied, we then say that $t_2 \in \mathcal{E}_j$, where

$$\mathcal{E}_j = \text{the set of external successful time steps for process } j.$$

The updates are

$$x_j^{t_2} = x_i^{t_1}$$

and

$$\Delta_j^{t_2} = \Delta_i^{t_1}.$$

We assume that the receipt of an external message does not affect the status of a function evaluation that may be executing on the receiving process.

3.2.3. Additional comments on what constitutes a success. Now that we have defined what constitutes both an internal and an external success, we define

$$\mathcal{S}_i = \mathcal{I}_i \cup \mathcal{E}_i = \text{the set of successful time steps for process } i.$$

We emphasize again that although internal successes require strict decrease in the function value as seen in (3.7), external successes relax the requirement of strict decrease and instead use the comparison operator $\prec$ to break ties, as shown in (3.11). This ensures that all processes agree on the best point even when different points generated by different processes have the same function value.

3.3. Unsuccessful time steps. Any time step that is not successful is classified as *unsuccessful*. We let the set

$$\mathcal{U}_i = \mathcal{T} \setminus \mathcal{S}_i$$

denote the unsuccessful time steps on process $i \in \mathcal{P}$. There are two types of unsuccessful time steps.

3.3.1. Contractions. Consider again the function evaluation on process i that starts at time step t_0 and finishes at time step t_1, as shown in Figure 3.1. We say that time step t_1 is a *contraction* if (3.7) is not satisfied *and* $x_i^{t_1-1} = x_i^{t_0}$; i.e., there is no reduction in the function value and x_i has not been updated since time step t_0 (which also means that $\Delta_i^{t_1-1} = \Delta_i^{t_0}$). In terms of time steps, $t_1 \notin \mathcal{I}_i$ and $t \notin \mathcal{E}_i$ for any $t \in \{t_0 + 1, \ldots, t_1 - 1\}$.

In this case, process i is required to reduce the value of its step-length control parameter $\Delta_i^{t_1-1}$ before continuing the search along its direction d_i. This means that $t \in \mathcal{T}_i$ since $\Delta_i^{t_1-1}$, though *not* $x_i^{t_1-1}$, is changed. More specifically, we say that $t_1 \in \mathcal{C}_i$, where

$$\mathcal{C}_i = \text{the set of contraction time steps for process } i.$$

Note that $\mathcal{S}_i \cap \mathcal{C}_i = \emptyset$ since $\mathcal{C}_i \subseteq \mathcal{U}_i$.

We update the step-length control parameter Δ_i as follows:

$$\Delta_i^{t_1} = \theta_i^{t_1} \Delta_i^{t_1-1},$$

where $\theta_i^{t_1}$ is the *contraction parameter* at time step t_1. The choice of the contraction parameter θ_i^t is subject to the following condition, using the same Λ as in (3.9):

$$\theta_i^t = \text{pow}(\Lambda, \ell_i^t) \tag{3.12}$$

for some

$$\ell_i^t \in \{-1, -2, -3, \ldots, \ell^{\min}\}, \tag{3.13}$$

where $\ell^{\min}$ is a finite integer constant. Together, (3.8), (3.12), and (3.13) imply that

$$\theta_i^t \in [\theta^{\min}, \theta^{\max}] \subset (0, 1), \quad \text{where } \theta^{\min} = \text{pow}(\Lambda, \ell^{\min}),\ \theta^{\max} = \text{pow}(\Lambda, -1). \tag{3.14}$$

3.3.2. The trivial case. The final possibility is that no changes to either x_i^t or Δ_i^t occur on process i for a given time step t; in other words, $t \notin \mathcal{T}_i$. This situation could occur for several reasons.

One possibility would be that process i is still evaluating f at a trial point constructed at some time step $t_0 < t$ and that evaluation does not finish during time step t. Thus, $t \notin \mathcal{I}_i$ and $t \notin \mathcal{C}_i$.

A further possibility is that no external candidate arrives from process j, $j \neq i$, or an external candidate does arrive, but it is immediately discarded since its function value does not improve upon $f(x_i^{t-1})$. Thus, $t \notin \mathcal{E}_i$.

A last possibility is that at time step t, process i does finish evaluating f at a trial point constructed at some time step $t_0 < t$ but the function value does not improve upon $f(x_i^{t-1})$, so $t \notin \mathcal{I}_i$. However, before assigning t to $\mathcal{C}_i$, we must verify that $x_i^{t-1} = x_i^t$. If $x_i^{t-1} \neq x_i^t$, that means that at least one external success occurred on process i at some time step $\hat{t} \in \{t_0+1, \ldots, t-1\}$. Let $\hat{t} = \max\{\{t_0+1, \ldots, t-1\} \cap \mathcal{E}_i\}$. In this case, since we have already recorded the external success at time step $\hat{t}$, we construct a new trial point without further changes to $x_i^{\hat{t}}$ or $\Delta_i^{\hat{t}}$ and initiate a new function evaluation. Thus, while $\hat{t} \in \mathcal{E}_i$, $t \in \mathcal{U}_i \setminus \mathcal{C}_i$.

3.4. Multiple decisions in one time step. We allow for the possibility that multiple candidates for the best point may be considered simultaneously at time step $t \in \mathcal{T}$ if, for instance, multiple messages have arrived from external processes or there is both an internal candidate as well as one or more external candidates to consider.

3.5. Identifying the source of a change. If a function evaluation finishes at time step t_1, a new one is started at time step t_1 using the values $x_i^{t_1}$ and $\Delta_i^{t_1}$—at least one of these values is guaranteed to have changed since time step t_0 from either an internal success, an external success, or a contraction.

To identify where a change to x_i^t, and possibly Δ_i^t, was generated (i.e., on which process) and at what time step the corresponding function evaluation started and finished, for each $i \in \mathcal{P}$ and for all $t \in \mathcal{S}_i$ we define the following *generating functions*:

$\omega_i(t) =$ the index of the process generating the update at time step t on process i,

$\tau_i(t) =$ the time index for the initiation of the function evaluation that produced the update at time step t on process i, and

$\nu_i(t) =$ the time index for the completion of the function evaluation that produced the update at time step t on process i.

Here

$$\omega_i(\cdot) : \mathcal{S}_i \to \mathcal{P}, \quad \tau_i(\cdot) : \mathcal{S}_i \to \mathcal{T}, \quad \nu_i(\cdot) : \mathcal{S}_i \to \mathcal{T}, \quad \text{and} \quad 0 \le \tau_i(t) < \nu_i(t) \le t.$$

For our example of an internal success on process i, so that $t_1 \in \mathcal{I}_i$, as illustrated in Figure 3.1, we have $\omega_i(t_1) = i$, $\tau_i(t_1) = t_0$, and $\nu_i(t_1) = t_1$. In fact, $\omega_i(t) = i$ and $\nu_i(t) = t$ for all $t \in \mathcal{I}_i$.

For our example of an external success on process j, so that $t_2 \in \mathcal{E}_j$, as illustrated in Figure 3.2, we have $\omega_j(t_2) = i$, $\tau_j(t_2) = \tau_i(t_1) = t_0$, and $\nu_j(t_2) = \nu_i(t_1) = t_1$.

The generating functions play an important role in the proofs of Lemma 5.1, Theorem 5.2, Lemma 7.4, and Corollary 7.5.

3.6. The definitions for x_i^t and Δ_i^t. For every $t \in \mathcal{T}$, $t > 0$, the best point x_i^t for process $i \in \mathcal{P}$ is defined to be

$$x_i^t = \begin{cases} x_{\omega_i(t)}^{\tau_i(t)} + \Delta_{\omega_i(t)}^{\tau_i(t)}\, d_{\omega_i(t)} & \text{if } t \in \mathcal{S}_i, \text{ and} \\ x_i^{t-1} & \text{otherwise.} \end{cases} \tag{3.15}$$

Recall that we initialize the procedure with x^0 as shown in (3.2). Thus, x_i^t is changed on process $i \in \mathcal{P}$ only at successful time steps $t \in \mathcal{S}_i$.

Changes in Δ_i^t must occur at contraction time steps and may occur at successful (internal or external) time steps. Correspondingly, for every $t \in \mathcal{T}$, $t > 0$, the step-length control parameter Δ_i^t for process $i \in \mathcal{P}$ is defined to be

$$\Delta_i^t = \begin{cases} \lambda_{\omega_i(t)}^{\nu_i(t)}\, \Delta_{\omega_i(t)}^{\tau_i(t)} & \text{if } t \in \mathcal{S}_i, \\ \theta_i^t \Delta_i^{t-1} & \text{if } t \in \mathcal{C}_i, \text{ and} \\ \Delta_i^{t-1} & \text{otherwise.} \end{cases} \tag{3.16}$$

Again, the initialization is as in (3.2), and we assume Δ^0 satisfies (3.4). Recall $\lambda_i^t \ge 1$ is the expansion parameter defined in (3.9) and $\theta_i^t \in (0, 1)$ is the contraction parameter defined in (3.12).

These precise definitions for x_i^t and Δ_i^t play a role in all the results that follow.

4. An overview of the analysis. Now that we have reviewed APPS and introduced most of the notation required for our analysis, we provide an outline of that analysis. Before proceeding, the reader may wish to review the example given in [6]. This example helps establish the definitions given in section 3, including those for the many sets we have introduced to track the progress of the search. Also, [6] illustrates and discusses those features of the asynchronous algorithm that make the analysis more intricate than for the synchronous case.

Our first task is to show that every iterate x_i^t lies on a rational lattice; this is equivalent to Theorem 3.2 in [10] for the synchronous case. The main difference here is that the asynchronism we have introduced in APPS complicates the arguments. Now, for some subset of the t's in $\mathcal{T}$, the x_i^t residing on process i may be the result of an external success—i.e., a point produced by a search along direction d_j on process j, where $j \neq i$. Thus the changes to x_i^t and Δ_i^t may be made without regard to the history of past successes on process i. Nevertheless, in section 5 we show that the algebraic structure found in the synchronous case is still preserved in the asynchronous case.

The lattice structure is the key to ensuring convergence for pattern search. In the synchronous case, the underlying lattice structure makes it possible to prove that a subsequence of the step-length control parameters goes to zero—even though pattern search does not enforce a sufficient decrease condition. In section 7, we show an equivalent result, but we now have p semi-independent sequences of Δ to consider. This makes the arguments more complex than in the synchronous case. Still, we arrive at Theorem 7.8, which says that

$$\liminf_{t \to +\infty} \Delta_j^t = 0 \quad \text{for all } j \in \mathcal{P}.$$

Our definition of Δ_i^t, given in (3.16), ensures that Δ_i^t is decreased only at contractions (i.e., when $t \in \mathcal{C}_i$). In fact, it is these contractions that are of interest for the remainder of the proof. In the synchronous case, there is an accumulation point $\hat{x}$ of the subsequence of iterates associated with contractions which has the property that $0 \leq f(\hat{x})^T d_i$ for all $i \in \mathcal{P}$. The challenge in the asynchronous case lies in showing that all the processes share a *common* accumulation point with this property. In order to do this, in section 8 we show that all the processes share a common subsequence of contraction iterates. This, in turn, relies on the fact that we require $\Delta^{\min} \leq \Delta$ every time we encounter a successful point, which guarantees that in the limit there will be long sequences of unsuccessful iterates. These long sequences of unsuccessful iterates allow us to construct a common sequence of contraction iterates across all processors. This argument culminates with Theorem 8.5, which states that there exists an $\hat{x}$ and, for each $i \in P$, a subset of the contraction iterates $\hat{\mathcal{C}}_i$ such that

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} \Delta_i^t = 0, \quad \lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} x_i^t = \hat{x}, \quad \text{and} \quad \lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} \nabla f(x_i^t) = \nabla f(\hat{x}) = 0 \quad \text{for all } i \in \mathcal{P}.$$

A fundamental difference in the assumptions for the asynchronous case is that the step-length control parameter is bounded below for all successful iterates (see (3.10)). This assumption is critical to the asynchronous case because it enables us to avoid a so-called race condition. If we did not enforce this lower bound, we might have different processes producing sequences of iterates that converge to different limit points, as the following situation in $\mathbb{R}^2$ illustrates. Let $f(x) = x^T x$. Observe that f is symmetric about both the x-axis and the y-axis and has a unique global minimizer

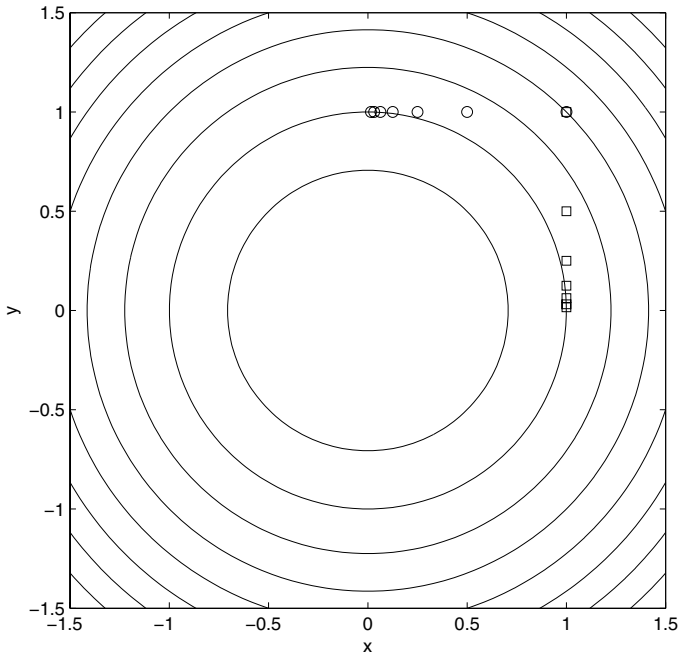

FIG. 4.1. *A potential race condition, which we exclude. In this illustration, if we do not enforce the bounds on* Δ_i^t *after an internal success, then each sequence converges to a different limit; the circles denote the sequence of best iterates on processes* 1 *and* 3 *while the squares denote the sequence of best iterates on process* 2. *The situation is remedied by requiring* $\Delta_i^t \geq \Delta^{\min}$ *for all* $t \in \mathcal{I}_i$.

at $x = (0,0)^T$. Choose

$$\mathcal{D} = \left\{ \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\},$$

which forms a positive basis for $\mathbb{R}^2$. Let $x^0 = (1,1)^T$, $\Delta^0 = 2$, and $\Lambda = 2$. For every contraction ($t \in \mathcal{C}_i$), choose $\theta = \Lambda^{-2} = 1/4$. For every internal success ($t \in \mathcal{I}_i$), choose $\lambda = \Lambda^1 = 2$, *ignoring* the lower bound restriction in (3.10). Assume further that each function evaluation requires exactly one time step and each communication takes exactly two time steps.

The situation that develops in this case is illustrated in Figure 4.1. Here, both $\{x_1^t\}$ and $\{x_3^t\} \to (0,1)^T$ (denoted by circles) while $\{x_2^t\} \to (1,0)^T$ (denoted by squares). The reason that the three sequences converge to two different limit points, neither of which is the unique stationary point for f, is that the first and the second processes are reducing their steps too quickly, continuing to find internal successes, and rejecting each candidate for an external success because they have already found another, better, internal success. The third process cannot remedy the situation since its direction is always a direction of ascent. The broadcasts of internal successes from the first and the second processes arrive on the third process within the same time step and have the same function value, which is better than any produced on the third process. Ties are broken in favor of the first process, leading to an external success on the third process, so the iterates on the third process also converge to $(0,1)^T$.

Enforcing the lower bound of $\Delta^{\min}$ on Δ_i^t for successful points eliminates this race condition. Choose, for instance, $\Delta^{\min} = 1/4$ and notice that this eventually disrupts the symmetric exchanges between the first and the second processes.

We require a *common* accumulation point $\hat{x}$ so that we can use the fact that our search directions in $\mathcal{D}$ form a positive spanning set, thus ensuring the final conclusion of Theorem 8.5: that $\hat{x}$ is also a stationary point of f.

We close by noting that in practice we stop searching along a given direction d_i once Δ_i^t falls below a certain threshold. Process i then waits until either another process reports a better point, in which case the search along d_i resumes with this new best point, or a sufficient number of other processes have converged to the same point identified by process i, in which case the entire search terminates. (For further details, see [5].) Thus, as a practical matter, $\Delta^{\min}$ need not impede the overall progress of the search, as it can be chosen to be on the order of the threshold used to terminate the search.

5. The algebraic structure of the iterates. We return to the analysis of APPS. We use the formulation for x_i^t given in (3.15) to show that we can, in fact, write any x_i^t as a linear combination of the search directions (translated by x^0). We prove this in Lemma 5.1. Then, in Theorem 5.2, we show that the algebraic structure underlying the sequences $\{x_i^t\}$, for all $i \in \mathcal{P}$, guarantees that all the iterates lie on a rational lattice defined by the search directions. The latter result is equivalent to Theorem 3.2 in [10].

LEMMA 5.1. *For any $i \in \mathcal{P}$ and any $t \in \mathcal{T}$, there exist sets $\hat{\mathcal{I}}_j(i,t) \subseteq \mathcal{I}_j$ for each $j \in \mathcal{P}$ such that*

$$x_i^t = x^0 + \sum_{j \in \mathcal{P}} \delta_j(i,t)\, d_j \quad \textit{with} \quad \delta_j(i,t) = \sum_{\hat{t} \in \hat{\mathcal{I}}_j(i,t)} \Delta_j^{\tau_j(\hat{t})}, \tag{5.1}$$

where $\delta_j(i,t) = 0$ if $\hat{\mathcal{I}}_j(i,t) = \emptyset$.

Proof. We prove this lemma by induction on t. For any $i \in \mathcal{P}$, the case for $t = 0$ is trivial since $x_i^0 = x^0$ by (3.2). Simply choose $\hat{\mathcal{I}}_j(i,0) = \emptyset$ for each $j \in \mathcal{P}$.

Now consider the case for general t for any $i \in \mathcal{P}$. First consider $t \in \mathcal{U}_i$, in which case (3.15) gives us $x_i^t = x_i^{t-1}$. From the induction hypothesis, we have

$$x_i^{t-1} = x^0 + \sum_{j \in \mathcal{P}} \delta_j(i,t-1)\, d_j \quad \text{with} \quad \delta_j(i,t-1) = \sum_{\hat{t} \in \hat{\mathcal{I}}_j(i,t-1)} \Delta_j^{\tau_j(\hat{t})}.$$

In this case, we simply choose $\hat{\mathcal{I}}_j(i,t) = \hat{\mathcal{I}}_j(i,t-1)$ for all $j \in \mathcal{P}$ to yield (5.1).

On the other hand, consider $t \in \mathcal{S}_i$. From (3.15), we have

$$x_i^t = x_{\omega_i(t)}^{\tau_i(t)} + \Delta_{\omega_i(t)}^{\tau_i(t)} d_{\omega_i(t)}.$$

The assumption that the minimum time for a function evaluation is one time step ensures that $\tau_i(t) < t$ for all $i \in \mathcal{P}$. Thus, from the induction hypothesis, we can rewrite the first term as

$$x_{\omega_i(t)}^{\tau_i(t)} = x^0 + \sum_{j \in \mathcal{P}} \delta_j(\omega_i(t), \tau_i(t))\, d_j \quad \text{with } \delta_j(\omega_i(t), \tau_i(t)) = \sum_{\hat{t} \in \hat{\mathcal{I}}_j(\omega_i(t), \tau_i(t))} \Delta_j^{\tau_j(\hat{t})}.$$

By definition, we also have $\tau_i(t) = \tau_{\omega_i(t)}(\nu_i(t))$ and $\nu_i(t) \in \mathcal{I}_{\omega_i(t)}$. Therefore, choosing

$$\hat{\mathcal{I}}_j(i,t) = \begin{cases} \hat{\mathcal{I}}_j(\omega_i(t), \tau_i(t)) \cup \{\nu_i(t)\} & \text{for } j = \omega_i(t), \text{ and} \\ \hat{\mathcal{I}}_j(\omega_i(t), \tau_i(t)) & \text{for } j \neq \omega_i(t) \end{cases}$$

yields (5.1). ∎

The purpose of the sets $\hat{\mathcal{I}}_j(i,t)$ is to track, for each $j \in \mathcal{P}$, which subset of the set of time steps that produced internal successes on process j led to the x_i^t residing on process i at time step t.

Now that we have taken a closer look at x_i^t, let us do the same for Δ_i^t. From (3.16), (3.12), and (3.9), we see that for any $i \in \mathcal{P}$ and for any $t \in \mathcal{T}$ we can express any Δ_i^t as a multiple of an integer power of the Λ from (3.8) times the Δ^0 from (3.4). Let Γ_i^t denote that integer power so that

$$\Delta_i^t = \text{pow}(\Lambda, \Gamma_i^t)\, \Delta^0, \quad \Gamma_i^t \in \mathbb{Z}. \tag{5.2}$$

Since $\Lambda \in \mathbb{Q}$, we can find Λ_{N} and Λ_{D} (here the subscripts denote *numerator* and *denominator*, respectively) such that

$$\Lambda = \frac{\Lambda_{\text{N}}}{\Lambda_{\text{D}}}, \quad \text{where } \Lambda_{\text{D}}, \Lambda_{\text{N}} \in \mathbb{N} \text{ with } \Lambda_{\text{D}}, \Lambda_{\text{N}} \text{ relatively prime.} \tag{5.3}$$

Using (5.3), we can rewrite (5.2) as

$$\Delta_i^t = \text{pow}(\Lambda_{\text{N}}, \Gamma_i^t)\ \text{pow}(\Lambda_{\text{D}}, -\Gamma_i^t)\, \Delta^0, \quad \Gamma_i^t \in \mathbb{Z}. \tag{5.4}$$

Define $\Gamma^{\max}$ to be the least integer such that

$$\text{pow}(\Lambda, \Gamma^{\max})\Delta^0 \geq \Delta^{\max}, \quad \Gamma^{\max} \in \mathbb{Z}. \tag{5.5}$$

From (3.10) we are then guaranteed that

$$\Gamma_i^t \leq \Gamma^{\max} \quad \text{for all } i \in \mathcal{P} \text{ and } t \in \mathcal{T}. \tag{5.6}$$

Finally, we recall the definition of the set of search directions $\mathcal{D}$ given in (2.1). Observe that each search direction $d_i \in \mathcal{D}$ is the product of a real nonsingular matrix B and a rational vector c_i.

Combining our observations on x_i^t and Δ_i^t, with our recollection of the definition of $\mathcal{D}$, we now state and prove the following theorem, which is our analogue of Theorem 3.2 from [10].

THEOREM 5.2. *Let $i \in \mathcal{P}$ and $\Gamma \in \mathbb{Z}$. For any $t \in \mathcal{T}$ such that*

$$\Gamma \leq \min\, \{\, \Gamma_i^{\tau_i(\hat{t})} \,:\, \hat{t} \leq t,\ \hat{t} \in \mathcal{I}_i,\ i \in \mathcal{P} \,\}, \tag{5.7}$$

where Γ_i^t is defined as in (5.2)*, there exists $\zeta_j(i,t,\Gamma) \in \mathbb{Z}$ for each $j \in \mathcal{P}$ such that*

$$x_i^t = x^0 + \frac{\text{pow}(\Lambda_{\text{N}}, \Gamma)}{\text{pow}(\Lambda_{\text{D}}, \Gamma^{\max})}\, \Delta^0\, B \sum_{j \in \mathcal{P}} \zeta_j(i,t,\Gamma)\, c_j, \tag{5.8}$$

where Λ_{N} and Λ_{D} are as defined in (5.3) *and $\Gamma^{\max}$ is as defined in* (5.5)*.*

Further, x_i^t lies on the rational lattice defined by integer multiples of the elements of the set $\{c_1, \ldots, c_p\}$ that is scaled by $\text{pow}(\Lambda_{\text{N}}, \Gamma)\ \text{pow}(\Lambda_{\text{D}}, -\Gamma^{\max})\, \Delta^0$*, translated by x^0, and subject to a (possible) change of basis B. This lattice is denoted by $\mathcal{G}(\mathcal{D}, \Lambda, \Gamma, \Delta^{\max}, \Delta^0, x^0)$.*

Proof. First we make an observation about any $\Delta_i^{\tau_i(\hat{t})}$ such that $i \in \mathcal{P}$, $\hat{t} \leq t$, and $\hat{t} \in \mathcal{I}_i$. From (5.4)–(5.6) we have

$$\Delta_i^{\tau_i(\hat{t})} = \text{pow}(\Lambda_\text{N}, \Gamma_i^{\tau_i(\hat{t})} - \Gamma)\ \text{pow}(\Lambda_\text{D}, \Gamma^\text{max} - \Gamma_i^{\tau_i(\hat{t})})\ \frac{\text{pow}(\Lambda_\text{N}, \Gamma)}{\text{pow}(\Lambda_\text{D}, \Gamma^\text{max})}\ \Delta^0.$$

Recall from (5.3) that Λ_D, $\Lambda_\text{N} \in \mathbb{N} \subset \mathbb{Z}$ and from (5.2) that $\Gamma_i^{\tau_i(\hat{t})} \in \mathbb{Z}$. Further, we have assumed that $\Gamma \in \mathbb{Z}$ and $\Gamma \leq \Gamma_i^{\tau_i(\hat{t})}$, and the assumptions placed on Γ^max ensure that $\Gamma^\text{max} \in \mathbb{Z}$ and $\Gamma^\text{max} \geq \Gamma_i^{\tau_i(\hat{t})}$. Combining these observations, we have that

$$\text{pow}(\Lambda_\text{N}, \Gamma_i^{\tau_i(\hat{t})} - \Gamma)\,\text{pow}(\Lambda_\text{D}, \Gamma^\text{max} - \Gamma_i^{\tau_i(\hat{t})}) \in \mathbb{Z}.$$

In Lemma 5.1 we saw that we could write any x_i^t as the sum of x^0 plus a linear combination of the search directions. Using the definitions of $\hat{\mathcal{I}}_j(i,t)$ and $\delta_j(i,t)$ from Lemma 5.1, we choose

$$\begin{aligned}\zeta_j(i,t,\Gamma) &= \sum_{\hat{t} \in \hat{\mathcal{I}}_j(i,t)} \text{pow}(\Lambda_\text{N}, \Gamma_j^{\tau_j(\hat{t})} - \Gamma)\,\text{pow}(\Lambda_\text{D}, \Gamma^\text{max} - \Gamma_j^{\tau_j(\hat{t})}) \\ &= \frac{\text{pow}(\Lambda_\text{D}, \Gamma^\text{max})\,\delta_j(i,t)}{\text{pow}(\Lambda_\text{N}, \Gamma)\,\Delta^0},\end{aligned}$$

with $\zeta_j(i,t,\Gamma) = 0$ if $\hat{\mathcal{I}}_j(i,t) = \emptyset$. Clearly, $\zeta_j(i,t,\Gamma) \in \mathbb{Z}$. Given that for every $j \in \mathcal{P}$, $d_j = Bc_j$, (5.8) then follows immediately from (5.1). The final statement follows from two facts. First, the set $\{c_1, \ldots, c_p\}$ is finite. Second, each of the c_j's is strictly rational and any finite set of rational numbers can be scaled to the integers. ∎

The importance of Theorem 5.2 will become apparent in Lemma 7.4, where we show that some subsequence of the step-length control parameters must go to zero.

6. The subset of time steps at which changes occur is infinite. Before we proceed to the proof of global convergence, we revisit the set $\mathcal{T}_i$, which we first defined in (3.1), and show that it must be infinite. A review of (3.15) and (3.16) leads to an alternate definition in terms of the subsets $\mathcal{S}_i$ and $\mathcal{C}_i$:

$$\mathcal{T}_i = \mathcal{S}_i \cup \mathcal{C}_i. \tag{6.1}$$

LEMMA 6.1. *$\mathcal{T}_i$ is infinite.*

Proof. Each function evaluation takes at most η time steps, and a new function evaluation is started at the conclusion of each function evaluation. Since $\mathcal{T}$ is infinite, there are infinitely many function evaluations. Recalling the discussion in section 3.5, for each function evaluation we are guaranteed that either an external successful update took place during the function evaluation or either an internal successful update or a contraction took place at the conclusion of the function evaluation. So, there must be at least one update to x_i and/or Δ_i for every function evaluation and, hence, there are infinitely many updates. ∎

This fact about $\mathcal{T}_i$ plays a role in the analysis ahead.

7. A subsequence of the step-length control parameters goes to zero. Once the lattice structure has been established, the next part of the proof of convergence for standard pattern search convergence analysis [10] involves showing that the step-length control parameter Δ goes to zero; i.e.,

$$\liminf_{t \to +\infty} \Delta^t = 0.$$

In this section, we aim to show an equivalent result, but we now have p semi-independent sequences of Δ to consider. Given this complication, the basic outline for our arguments is as follows:

1. If the number of successful time steps for some process is finite, showing that the sequence of step-length control parameters goes to zero is trivial. So, we eliminate this case first in Lemma 7.1.

2. Using Lemma 7.1, we then show, in Lemma 7.2 and Corollary 7.3, that either every process has a set of successful time steps that is finite or none do. From this point forward, we then need only concern ourselves with the case where the number of successful time steps is infinite.

3. Lemma 7.4 is a key result. We show that some subsequence of the set of all step-length control parameters (indexed over all processes and all time steps) must go to zero. This result relies on the fact that every x_i^t lies on a rational lattice.

4. We narrow the scope in Corollary 7.5 to show that a subsequence of step-length control parameters converges to zero on *one* process $i \in \mathcal{P}$.

5. Before we can extend this result to the remaining processes, we introduce some new definitions that help us discover what is happening between successful time steps on any process $j \in \mathcal{P}$, $j \neq i$. In Lemma 7.6, we conclude that the lim sup of the number of time steps between successes on a single process goes to $+\infty$ in these cases.

6. We now can tie together the actions across processes to say, in Lemma 7.7, that *every* process must have a subsequence of step-length control parameters that goes to zero.

7. Combining all these results into Theorem 7.8, we see that whether or not the number of successful time steps is infinite, every process has a subsequence of step-length control parameters that goes to zero.

Now that we have an overall picture of the argument, we begin by showing that for any process i which has only finitely many successful time steps, the sequence of step-length control parameters goes to zero.

LEMMA 7.1. *If $\mathcal{S}_i$ is finite for some $i \in \mathcal{P}$, then*

$$\lim_{t \to +\infty} \Delta_i^t = 0.$$

Proof. Let $t_0 = \max \{ t : t \in \mathcal{S}_i \}$. Then, by (3.16), for any time step $t \in \mathcal{T}$ such that $t > t_0$, either the time step is a contraction or nothing happens. From (6.1), we have $\mathcal{T}_i = \mathcal{S}_i \cup \mathcal{C}_i$, and Lemma 6.1 assures us that $\mathcal{T}_i$ is infinite. Since, by assumption, the set $\mathcal{S}_i$ is finite, we conclude that the set $\mathcal{C}_i$ must be infinite. Hence there are infinitely many contractions after time step t_0. Therefore, the sequence $\{\Delta_i^t\}_{t=t_0}^{+\infty}$ is decreasing and bounded below by zero. Finally, (3.14) guarantees that the contraction parameter $\theta_i^t \leq \theta^{\max} < 1$, which enforces a fraction of decrease at each contraction. We can therefore conclude that the sequence $\{\Delta_i^t\}_{t=t_0}^{+\infty}$ converges to zero. Hence, the claim. ∎

In the next lemma, we show that if one process has infinitely many successful time steps, then *every* process must have infinitely many successful time steps.

LEMMA 7.2. *If $\mathcal{S}_i$ is infinite for some $i \in \mathcal{P}$, then $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$.*

Proof. Suppose not; that is, suppose there exists $k \in \mathcal{P}$, $k \neq i$, such that $\mathcal{S}_k$ is finite. Let $t_0 = \max \{ t : t \in \mathcal{S}_k \}$, which implies that $x_k^{t_0}$ is the best point known by process k over all $t \in \mathcal{T}$. The point $x_k^{t_0}$ is considered by process i at some later time step $t_1 \leq t_0 + \gamma$, where γ is defined in (3.6). Since S_i is infinite, $x_k^{t_0}$, whether initially accepted or rejected at time step t_1, is improved upon at some later time step

t_2 with $t_2 > t_1$; together, (3.5) and (3.6) guarantee that t_2 is finite. The point $x_i^{t_2}$ must, in turn, be considered by process k at a later time step $t_3 \leq t_2 + \gamma$. Since $x_i^{t_2}$ is an improvement over $x_k^{t_0}$, we must have $t_3 \in \mathcal{S}_k$; but this contradicts t_0 being the maximum $t \in \mathcal{S}_k$. ∎

The immediate corollary is that if any process has only finitely many successful time steps, then *every* process has only finitely many successful time steps.

COROLLARY 7.3. *If $\mathcal{S}_i$ is finite for some $i \in \mathcal{P}$, then $\mathcal{S}_j$ is finite for all $j \in \mathcal{P}$.*

From Lemma 7.1 and Corollary 7.3, the case for the convergence of the step-length control parameters to zero is trivial when there are finitely many successful time steps. The remainder of this section concentrates on the case where there are infinitely many successful time steps on each process.

The next lemma shows there is a subsequence of step-length control parameters (indexed over all processes) that converges to zero.

LEMMA 7.4. *Suppose $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$. Then there exists $i \in \mathcal{P}$ such that*

$$\liminf_{\substack{t \to +\infty \\ t \in \mathcal{S}_i}} \Delta_{\omega_i(t)}^{\tau_i(t)} = 0.$$

Proof. Suppose not. Then there exists $\Delta^* > 0$ such that

$$\Delta_{\omega_j(t)}^{\tau_j(t)} \geq \Delta^* \quad \text{for all } j \in \mathcal{P} \text{ and } t \in \mathcal{S}_j.$$

Choose a $\Gamma^* \in \mathbb{Z}$ that satisfies (5.7) for all $t \in \mathcal{T}$. We are guaranteed that such a Γ^* exists since Δ^* is strictly positive. With this choice of Γ^*, Theorem 5.2 guarantees that (5.8) holds for *every* choice of $t \in \mathcal{T}$, and thus every x_j^t lies on the translated rational lattice $\mathcal{G}(\mathcal{D}, \Lambda, \Gamma^*, \Delta^{\max}, \Delta^0, x^0)$.

Observe that each lattice point in $\mathcal{G}(\mathcal{D}, \Lambda, \Gamma^*, \Delta^{\max}, \Delta^0, x^0)$ can be considered successful at most once by each process. Consider process $k \in \mathcal{P}$. Recall that $\mathcal{S}_k = \mathcal{I}_k \cup \mathcal{E}_k$ and a successful point must satisfy either (3.7) or (3.11). In either case, if $f(x_k^{t_2}) < f(x_k^{t_1})$, then clearly $x_k^{t_1} \neq x_k^{t_2}$. The only other possibility is that $t_2 \in \mathcal{E}_k$ with $f(x_k^{t_2}) = f(x_k^{t_1})$, in which case we must have $x_k^{t_2} \prec x_k^{t_1}$ so that, once again, $x_k^{t_1} \neq x_k^{t_2}$. We conclude, therefore, that for any process $k \in \mathcal{P}$, we cannot have $t_1, t_2 \in \mathcal{S}_k$ with $t_1 < t_2$ such that $x_k^{t_1} = x_k^{t_2}$.

On the other hand, every successful point must lie in $\mathcal{L}(x^0)$. The intersection of the bounded set $\mathcal{L}(x^0)$ with the translated integer lattice $\mathcal{G}(\mathcal{D}, \Lambda, \Gamma^*, \Delta^{\max}, \Delta^0, x^0)$ is finite.

Since any successful point must be in the finite set $\mathcal{G}(\mathcal{D}, \Lambda, \Gamma^*, \Delta^{\max}, \Delta^0, x^0) \cap \mathcal{L}(x^0)$ and no point is successful more than once for each process $j \in \mathcal{P}$, it follows that $\mathcal{S}_j$ must be finite. But this contradicts the assumption that $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$. Hence, the claim. ∎

An immediate corollary to the preceding lemma is that there is some process which has a subsequence of step-length control parameters that converges to zero.

COROLLARY 7.5. *Suppose $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$. Then there exists $i \in \mathcal{P}$ such that*

$$\liminf_{t \to +\infty} \Delta_i^t = 0. \tag{7.1}$$

Proof. By Lemma 7.4, there exists $i \in \mathcal{P}$ and $\bar{\mathcal{S}}_i \subseteq \mathcal{S}_i$ such that

$$\lim_{\substack{t \to +\infty \\ t \in \bar{\mathcal{S}}_i}} \Delta_{\omega_i(t)}^{\tau_i(t)} = 0.$$

For each $j \in \mathcal{P}$, define $\bar{\mathcal{S}}_{ij} = \{t \in \bar{\mathcal{S}}_i : \omega_i(t) = j\}$, so $\bigcup_{j=1}^{p} \bar{\mathcal{S}}_{ij} = \bar{\mathcal{S}}_i$. Since $\bar{\mathcal{S}}_i$ is infinite, there exists at least one k such that $\bar{\mathcal{S}}_{ik}$ is infinite. So,

$$\lim_{\substack{t \to +\infty \\ t \in \bar{\mathcal{S}}_{ik}}} \Delta_k^{\tau_i(t)} = 0.$$

Hence, the claim. ∎

We need to show that a subsequence of step-length control parameters is going to zero for *every* process. In order to do so, we must first introduce some definitions and an additional lemma.

For each process $i \in \mathcal{P}$, we can decompose the set of unsuccessful time steps (i.e., $t \notin \mathcal{S}_i$) into contiguous blocks as follows:

$$\mathcal{U}_i = \mathcal{T} \setminus \mathcal{S}_i = \mathcal{U}_{i1} \cup \mathcal{U}_{i2} \cup \cdots \cup \mathcal{U}_{iN}, \tag{7.2}$$

where N may be $+\infty$, each $\mathcal{U}_{i\ell}$ is a contiguous index block (e.g., $\mathcal{U}_{i\ell} = \{3, 4, 5, 6\}$), and any pair $\mathcal{U}_{i\ell}$ and $\mathcal{U}_{i,\ell+1}$ is separated by at least one $t \in \mathcal{S}_i$.

It is also useful to define the minimum number of contractions required to reduce $\Delta^{\min}$ to a given $\Delta \in \mathbb{R}$, $\Delta > 0$, as

$$\underline{\kappa}(\Delta) = \min \{ p \in \{0, 1, 2, \dots\} : \operatorname{pow}(\theta^{\min}, p)\, \Delta^{\min} \leq \Delta \}, \tag{7.3}$$

where $\theta^{\min}$ is defined in (3.14) and $\Delta^{\min}$ is defined in (3.10). It is straightforward to see that

$$\lim_{\Delta \to 0} \underline{\kappa}(\Delta) = +\infty. \tag{7.4}$$

Finally, for a given $t \in \mathcal{T}$, we define the last successful time step up to, and possibly including, t and the first successful time step after t as

$$\psi_i(t) = \max \{ \bar{t} \in \mathcal{S}_i \cup \{0\} : \bar{t} \leq t \}, \quad \text{and} \tag{7.5}$$

$$\phi_i(t) = \min \{ \bar{t} \in \mathcal{S}_i : t < \bar{t} \}, \tag{7.6}$$

respectively. We ensure that $\psi_i(t)$ is always defined by setting it to zero in the case that $\{ \bar{t} \in \mathcal{S}_i : \bar{t} \leq t \}$ is empty. In the case that there is no $\bar{t} \in \mathcal{S}_i$ satisfying $t < \bar{t}$, then $\phi_i(t) = +\infty$. Thus, $\psi_i(\cdot) : \mathcal{T} \to \mathcal{S}_i \cup \{0\}$, $\phi_i(\cdot) : \mathcal{T} \to \mathcal{S}_i \cup \{+\infty\}$, and $\psi_i(t) < \phi_i(t)$ for all $t \in \mathcal{T}$.

Using the above definitions, we can show that the lim sup of the number of time steps between successes is going to infinity if a subsequence of the step-length control parameters is going to zero.

LEMMA 7.6. *Suppose $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$. Then for all $i \in \mathcal{P}$ satisfying* (7.1)*, we have*

$$\limsup_{\ell \to +\infty} |\mathcal{U}_{i\ell}| = +\infty. \tag{7.7}$$

Proof. Let $i \in \mathcal{P}$ be such that (7.1) holds. By the definition of the limit, for any $\Delta^* > 0$, there exists $t^* \in \mathcal{T}$ such that $\Delta_i^{t^*} < \Delta^*$. Without loss of generality, we assume $t^* \in \mathcal{U}_i$.

Then, using definitions (7.3) and (7.5) from above, there must be at least $\underline{\kappa}(\Delta^*)$ time steps between t^* and $\psi_i(t^*)$ since (3.10) must hold for all $t \in \mathcal{S}_i$. Let ℓ^* be such that $t^* \in \mathcal{U}_{i\ell^*}$. Then

$$|\mathcal{U}_{i\ell^*}| > \underline{\kappa}(\Delta^*).$$

From (7.4), the proof is complete. ∎

We can now show that, in the case of an infinite number of successful time steps, a subsequence of the step-length control parameters converges to zero for every process.

LEMMA 7.7. *Suppose $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$. Then for all $j \in \mathcal{P}$,*

$$\liminf_{t \to +\infty} \Delta_j^t = 0.$$

Proof. Suppose not. Then there exists an $i \in \mathcal{P}$ and $\Delta^* > 0$ such that

$$\Delta_i^t \geq \Delta^* \quad \text{for all } t \in \mathcal{T}.$$

Define

$$\bar{\kappa}(\Delta^*) = \min\{p \in \{0, 1, 2, \dots\} : \text{pow}(\theta^{\max}, p)\Delta^{\max} \leq \Delta^*\},$$

where $\theta^{\max}$ is defined in (3.14) and $\Delta^{\max}$ is defined in (3.10). Then $\bar{\kappa}(\Delta^*)$ is the maximum possible number of contractions needed to reduce $\Delta^{\max}$ to Δ^*. So the maximum number of time steps between two successful time steps on process i is

$$\max_{\ell} |\mathcal{U}_{i\ell}| \leq \eta\, \bar{\kappa}(\Delta^*),$$

where η is defined in (3.5) and $\mathcal{U}_{i\ell}$ is defined in (7.2).

Now consider $k \in \mathcal{P}$, $k \neq i$. Since any successful point produced on process k is considered on process i within γ time steps, i has a new minimum within $\eta\, \bar{\kappa}(\Delta^*)$ time steps, and that new minimum is considered by process k within γ more time steps; so the maximum number of time steps between successes on any process k, $k \neq i$, can be at most

$$\max_{\ell} |\mathcal{U}_{k\ell}| \leq \eta\, \bar{\kappa}(\Delta^*) + 2\gamma. \tag{7.8}$$

However, Corollary 7.5 guarantees us that there exists i^* such that (7.1) holds, and our null hypothesis tells us $i^* \neq i$. Further, Lemma 7.6 says (7.7) must hold for i^*, but this contradicts (7.8), which also holds for $k = i^*$. Hence, the claim. ∎

Finally, we show that each process has a subsequence of step-length control parameters that converges to zero—whether there are finitely or infinitely many successful time steps.

THEOREM 7.8. *For every process $j \in \mathcal{P}$, there exists a subsequence of the step-length control parameters that goes to zero; that is,*

$$\liminf_{t \to +\infty} \Delta_j^t = 0 \quad \textit{for all } j \in \mathcal{P}.$$

Proof. If $\mathcal{S}_i$ is infinite for some $i \in \mathcal{P}$, then $\mathcal{S}_j$ is infinite for all $j \in \mathcal{P}$ by Lemma 7.2, in which case the claim follows immediately from Lemma 7.7. Otherwise, all $\mathcal{S}_j$ must be finite for all $j \in \mathcal{P}$ by Corollary 7.3, in which case the claim follows from Lemma 7.1. ∎

The following corollary says that, specifically, the subsequence of time steps at which the step-length control parameters decrease forms a subset of the set of unsuccessful time steps. This corollary is useful in the next section.

COROLLARY 7.9. *The set $\mathcal{C}_j$ is infinite for all $j \in \mathcal{P}$, and*

$$\liminf_{\substack{t \to +\infty \\ t \in \mathcal{C}_j}} \Delta_j^t = 0 \quad \textit{for all } j \in \mathcal{P}. \tag{7.9}$$

Proof. This follows immediately from Theorem 7.8 since for each $j \in \mathcal{P}$, $\Delta_j^t \geq \Delta^{\min}$ for all $t \in \mathcal{S}_j$ and (3.16) confirms that Δ_j^t is unchanged for all $t \in \mathcal{T} \setminus \mathcal{T}_i$. ∎

8. A common accumulation point that is also a stationary point. Our final goal is to show that there exists a common accumulation point for all processes and that this accumulation point has a zero gradient. Our argument is outlined as follows.

1. In Lemma 8.1 we show that on process 1 we can extract a subsequence of contractions for which the step-length control parameter goes to zero and that the subsequence x_1^t associated with these particular contractions has an accumulation point. (We specify the first process for convenience, but we could pick any process.)

2. Still focused on process 1, in Corollary 8.2 we show that the number of unsuccessful time steps before each of these contractions is going to $+\infty$. This means that on process 1 we have a sequence of ever-lengthening contiguous index blocks of unsuccessful time steps.

3. In Lemma 8.3 we show that using the subsequence of contractions on process 1 for which the step-length control parameter goes to zero, each process i, $i \neq 1$, has its own corresponding sequence of contiguous index blocks of unsuccessful time steps.

4. In Lemma 8.4, we show that the sequence of contiguous index blocks of unsuccessful time steps on each process i, $i \neq 1$, is also ever-lengthening. We then extract a sequence of step-length control parameters corresponding to these ever-lengthening blocks of unsuccessful time steps and show that this particular sequence of step-length control parameters must go to zero.

5. Finally, in Theorem 8.5 we show that these blocks of unsuccessful iterates can be used to build a sequence of contraction iterates corresponding to those on process 1—and thus share the same accumulation point. Furthermore, if we use the fact that the set of search directions positively spans $\mathbb{R}^n$, and assume that f is continuously differentiable, then we can show that this common accumulation point is also a stationary point of f.

In essence, our argument for the existence of a common accumulation point is based on the timing of the global clock. Since we have assumed both that the number of time steps required for a function evaluation is finite (3.5) and that the number of time steps required for the communication of a message is finite (3.6), we know that eventually every process must see any candidate for the new best point in a finite amount of time. What we do not know, in general, is in what order each candidate will be considered on any given process. What we show is that there is an infinite sequence of increasingly long blocks of unsuccessful time steps on every process, where the block length is unbounded above as the algorithm proceeds. We also show that every sufficiently long block is a member of a set of such blocks, where all the blocks in a set have start times within γ time steps of one another. Similarly, the same can be said for all finish times. We then show that each set contains one block for each process. For a set of sufficiently long blocks, each process must start and finish a function evaluation within that process's block. The bounds (3.5) and (3.6) mean that all processes start these new function evaluations using the *same* best point. For sufficiently long blocks, all of these function evaluations must be unsuccessful. Thus, in the language of [10], the processes collectively perform a poll about the best point, and this poll is unsuccessful. The sequence of such sets of blocks is infinite, and so an infinite sequence of these best points occurs. The final conclusion, that this accumulation point is also a stationary point of f, follows automatically from our assumptions on $\mathcal{D}$ and f.

Having made these observations, we start the analysis by showing that the first process has a convergent subsequence of x's corresponding to a subsequence of step-length control parameters that goes to zero.

LEMMA 8.1. *There exists* $\hat{x} \in \mathbb{R}^n$ *and* $\hat{\mathcal{C}}_1 \subseteq \mathcal{C}_1$ *such that*

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_1}} \Delta_1^t = 0 \qquad \text{and} \qquad \lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_1}} x_1^t = \hat{x}. \tag{8.1}$$

Proof. From Corollary 7.9, we know that $\mathcal{C}_1$ is infinite and that (7.9) holds, so there exists $\mathcal{C}_1' \subseteq \mathcal{C}_1$ such that

$$\lim_{\substack{t \to +\infty \\ t \in \mathcal{C}_1'}} \Delta_1^t = 0.$$

Since the set $\{ x_1^t : t \in \mathcal{C}_1' \}$ is contained in the bounded set $\mathcal{L}(x^0)$, we can extract an infinite subset $\hat{\mathcal{C}}_1 \subset \mathcal{C}_1'$ such that the subsequence converges; i.e., there exists $\hat{x}$ in the closure of $\mathcal{L}(x^0)$ such that the limit in (8.1) holds. ∎

Next, we show that the number of time steps between each $t \in \hat{\mathcal{C}}_1$ and the most recent success on process 1 goes to $+\infty$.

COROLLARY 8.2. *Let* $\hat{\mathcal{C}}_1$ *be as defined in Lemma* 8.1. *Then there exists* $t^* \in \mathcal{T}$ *such that*

$$\underline{\kappa}(\Delta_1^t) > \eta + 2\gamma \quad \text{for all } t > t^*,\ t \in \hat{\mathcal{C}}_1,$$

where $\underline{\kappa}(\Delta)$ *is defined in* (7.3), η *is defined in* (3.5), *and* γ *is defined in* (3.6).

Proof. This follows immediately from Lemma 8.1 and (7.4). ∎

Another way to look at this corollary is to consider the step-length control parameters. By definition, $\underline{\kappa}(\Delta)$ returns the minimum number of contractions required to reduce $\Delta^{\min}$ to a given value Δ. Consider $\hat{t} > t^*$ with $\hat{t} \in \hat{\mathcal{C}}_1$. Corollary 8.2 then tells us that $\underline{\kappa}(\Delta_1^t)$ is at least $\eta + 2\gamma$. The importance of this connection with $\Delta^{\min}$ becomes clearer when we recall that (3.10) requires the Δ associated with any successful time step to satisfy $\Delta_i^t \geq \Delta^{\min}$. Therefore, we conclude that the minimum possible number of contractions since the last successful time step, at time step $\psi_1(\hat{t}\,)$, is $\eta + 2\gamma$. Since each contraction requires one function evaluation which, in turn, requires at least one time step, the situation illustrated in Figure 8.1 must hold.

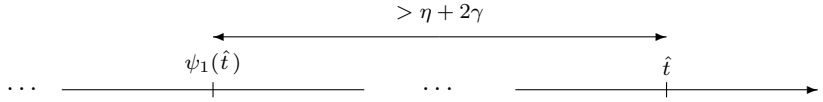

FIG. 8.1. *Relative order of events on process* 1 *when* $\hat{t} \in \hat{\mathcal{C}}_1$ *and* $\hat{t} > t^*$.

The situation illustrated in Figure 8.1 applies only to process 1. Now we show that for every $\hat{t} \in \hat{\mathcal{C}}_1$, $\hat{t} > t^*$, on each of the other processes there is a corresponding nonempty block of contiguous time steps that is devoid of successes. In particular, the situation shown in Figure 8.2 holds. The relative order between the time steps $\psi_1(\hat{t}\,) + \gamma$ and $\hat{t} - \gamma$ follows from Corollary 8.2. In the next lemma, we show that the relative order of the time steps $\psi_i(\hat{t} - \gamma)$ and $\psi_1(t) + \gamma$, as well as that of the time steps $\hat{t} - \gamma$ and $\phi_i(\psi_1(\hat{t}\,) + \gamma)$, also must hold for any $i \in \mathcal{P}$, $i \neq 1$, when $\hat{t} \in \hat{\mathcal{C}}_1$ and $\hat{t} > t^*$. The result we want then follows immediately.

LEMMA 8.3. *Let* $\hat{\mathcal{C}}_1$ *be as defined in Lemma* 8.1 *and let* t^* *be as defined in Corollary* 8.2. *Then for any* $\hat{t} \in \hat{\mathcal{C}}_1$ *with* $\hat{t} > t^*$ *and any* $i \in \mathcal{P}$, $i \neq 1$, *we have*

$$\psi_i(\hat{t} - \gamma) \leq \psi_1(\hat{t}\,) + \gamma \qquad \text{and} \tag{8.2}$$

$$\hat{t} - \gamma \leq \phi_i(\psi_1(\hat{t}\,) + \gamma), \tag{8.3}$$

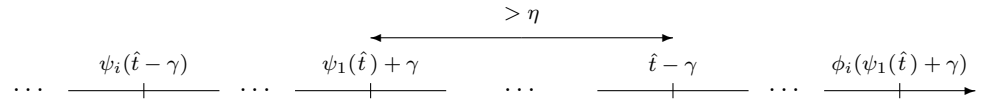

FIG. 8.2. *Relative order of events for any process $i \in \mathcal{P}$, $i \neq 1$, when $\hat{t} \in \hat{\mathcal{C}}_1$ and $\hat{t} > t^*$.*

where γ is defined in (3.6)*, $\psi_i(\cdot)$ is defined in* (7.5)*, and $\phi_i(\cdot)$ is defined in* (7.6)*. Further,*

$$\{\, t \in \mathcal{T} \,:\, \psi_1(\hat{t}\,) + \gamma < t < \hat{t} - \gamma \,\} \;\subseteq\; \mathcal{U}_i, \tag{8.4}$$

where $\mathcal{U}_i$ is defined in (7.2)*.*

Proof. Suppose not. First consider the proof for (8.2). Since the point $x_1^{\psi_1(\hat{t}\,)}$ is guaranteed to have been considered by process i by time step $\psi_1(\hat{t}\,)+\gamma$ and $\psi_1(\hat{t}\,)+\gamma < \psi_i(\hat{t}-\gamma)$ (from the null hypothesis), it must be true that

$$f(x_i^{\psi_i(\hat{t}-\gamma)}) < f(x_1^{\psi_1(\hat{t}\,)}), \tag{8.5}$$

or, equivalently for our purposes, that the tie-breaking condition in (3.11) is satisfied. Likewise, the point $x_i^{\psi_i(\hat{t}-\gamma)}$ will be considered by process 1 at some time step $t_1 \geq \psi_i(\hat{t}-\gamma)$. By the null hypothesis, we have $\psi_1(\hat{t}\,) < \psi_i(\hat{t}-\gamma)-\gamma$, so $\psi_1(\hat{t}\,) < t_1$. On the other hand, since the point $x_i^{\psi_i(\hat{t}-\gamma)}$ must be considered within γ time steps of $\psi_i(\hat{t}-\gamma)$, we have $t_1 \leq \psi_i(\hat{t}-\gamma)+\gamma$. By the definition of ψ, we conclude $t_1 \leq \hat{t}$. So we then have

$$\psi_1(\hat{t}\,) < t_1 \leq \hat{t}.$$

From (8.5), either $t_1 \in \mathcal{S}_1$, or there exists $t_2 \in \mathcal{S}_1$ with $\psi_1(\hat{t}\,) < t_2 < t_1$. In either case, we have a contradiction to the fact that $\psi_1(\hat{t}\,)$ is the most recent successful time step before $\hat{t}$ on process 1.

We follow the same line of reasoning for (8.3). Since $\phi_i(\psi_1(\hat{t}\,)+\gamma) \in \mathcal{S}_i$ (note that it is finite by the null hypothesis) and the point $x_1^{\psi_1(\hat{t}\,)}$ must have been considered by time step $\psi_1(\hat{t}\,)+\gamma$, it must be true that

$$f(x_i^{\phi_i(\psi_1(\hat{t}\,)+\gamma)}) < f(x_1^{\psi_1(\hat{t}\,)}), \tag{8.6}$$

or, equivalently for our purposes, that the tie-breaking condition in (3.11) is satisfied. Likewise, the point $x_i^{\phi_i(\psi_1(\hat{t}\,)+\gamma)}$ will be considered by process 1 by some time step t_1 satisfying

$$\psi_1(\hat{t}\,) < \phi_i(\psi_1(\hat{t}\,)+\gamma)-\gamma \leq t_1 \leq \phi_i(\psi_1(\hat{t}\,)+\gamma)+\gamma < \hat{t},$$

where the last part is from the null hypothesis and the first part is from the definition of ϕ. From (8.6), either $t_1 \in \mathcal{S}_1$ or there exists $t_2 \in \mathcal{S}_1$ with $\psi_1(\hat{t}\,) < t_2 < t_1$. In either case, we once again have a contradiction.

The proof for (8.4) follows immediately. ∎

Using the previous lemma, we can construct a set of time steps $\hat{\mathcal{C}}_i$ such that the corresponding sequence of step-length control parameters converges to zero.

LEMMA 8.4. *Consider any $i \in \mathcal{P}$, $i \neq 1$. Let $\hat{\mathcal{C}}_1$ be as defined in Lemma* 8.1 *and let t^* be as defined in Corollary* 8.2*. For any $\hat{t} \in \hat{\mathcal{C}}_1$ with $\hat{t} > t^*$ define*

$$\chi_i(\hat{t}\,) \;=\; \max\{\, t \in \mathcal{C}_i \,:\, t < \hat{t} - \gamma \,\} \tag{8.7}$$

and

$$\hat{\mathcal{C}}_i = \{\, \chi_i(\hat{t}\,) \,:\, \hat{t} > t^*,\ \hat{t} \in \hat{\mathcal{C}}_1 \,\}.$$

Then

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} \Delta_i^t = 0. \tag{8.8}$$

Proof. First, we are guaranteed that

$$\chi_i(\hat{t}\,) > \psi_1(\hat{t}\,) + \gamma \quad \text{for each } \hat{t} \in \hat{\mathcal{C}}_1 \text{ with } \hat{t} > t^*$$

for the following reasons. Appealing to Corollary 8.2, we know $\underline{\kappa}(\Delta_1^{\hat{t}}) > \eta + 2\gamma$ and so the interval contains at least η time steps. Thus, one function evaluation must finish and another start on process i during that interval. Since, by Lemma 8.3, there are no successes on i between $\psi_1(\hat{t}\,) + \gamma$ and $\hat{t} - \gamma$, there must be at least one contraction on i in that interval, i.e., a $t \in \mathcal{C}_i$.

Next, from Lemma 8.1 and (7.4), we know that

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_1}} \underline{\kappa}(\Delta_1^t) = +\infty,$$

so it must also be the case that

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_1}} \frac{(t - \gamma) - (\psi_1(t) + \gamma)}{\eta} = +\infty.$$

In other words, the number of contractions in the interval defined by (8.4) is tending towards infinity. Therefore, (8.8) holds. ∎

Finally, we conclude that all processes share a common accumulation point and that such a point is a stationary point of f. This argument follows the same basic lines as those seen in [3, 9] (for the case that the search directions are restricted to the set $\mathcal{D} = \{\pm e_i, i = 1, \ldots, n\}$) and [11] (for the general case that $\mathcal{D}$ is a positive spanning set). Similar arguments have been used more recently in [8, 1, 4].

THEOREM 8.5. *Assume the function f in* (1.1) *is continuously differentiable on the closure of $\mathcal{L}(x^0)$. Then there exists $\hat{x} \in \mathbb{R}^n$ and, for each $i \in \mathcal{P}$, there exists $\hat{\mathcal{C}}_i \subset \mathcal{C}_i$ such that*

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} \Delta_i^t = 0 \qquad \textit{and} \qquad \lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} x_i^t = \hat{x}. \tag{8.9}$$

Furthermore,

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{C}}_i}} \nabla f(x_i^t) = 0.$$

Proof. By Lemma 8.1, we know that (8.9) holds for $i = 1$. By Lemma 8.4, we know that for each $i \in \mathcal{P}$, $i \neq 1$, we can construct $\hat{\mathcal{C}}_i$ such that the limit on Δ_i^t in (8.9) holds. Further, note that for every $\hat{t} \in \hat{\mathcal{C}}_i$, we have

$$x_i^{\chi_i(\hat{t}\,)} = x_1^{\hat{t}},$$

where $\chi_i(\hat{t}\,)$ is defined in (8.7). Thus,

$$\{\, x_i^t \,:\, t \in \hat{\mathcal{C}}_i \,\} \subseteq \{\, x_1^t \,:\, t \in \hat{\mathcal{C}}_1 \,\}.$$

So, the limit on x_i^t given in (8.9) holds as well. Hence, the claim.

Now, for any $t \in \mathcal{C}_i$, (3.15) and (3.16) give us

$$x_i^t = x_i^{t-1} \qquad \text{and} \qquad \Delta_i^t = \theta_i^t \Delta_i^{t-1}.$$

Define the set $\hat{\mathcal{B}}_i = \{\, t = \hat{t} - 1 \,:\, \hat{t} \in \hat{\mathcal{C}}_i \,\}$. Since θ_i^t is bounded below by $\theta^{\min}$, (8.9) ensures that

$$\lim_{\substack{t \to +\infty \\ t \in \hat{\mathcal{B}}_i}} \Delta_i^t = 0.$$

If $\hat{t} \in \hat{\mathcal{C}}_i$ this means that

$$f(x_i^{\hat{t}-1}) \leq f(x_i^{\hat{t}-1} + \Delta_i^{\hat{t}-1} d_i). \tag{8.10}$$

We rely here on the fact that even though the function evaluation that led to the conclusion that $\hat{t} \in \hat{\mathcal{C}}_i$ may have been initiated at some $t < \hat{t} - 1$, the update rules (3.15) and (3.16) ensure that $x_i^t = x_i^{t-1}$ and $\Delta_i^t = \Delta_i^{t-1}$ for any $t \in \mathcal{T} \setminus \mathcal{T}_i$. Since (8.10) holds for any $\hat{t} \in \hat{\mathcal{C}}_i$, this is equivalent to saying that for any $t \in \hat{\mathcal{B}}_i$

$$f(x_i^t) \leq f(x_i^t + \Delta_i^t d_i).$$

The mean value theorem then gives us

$$f(x_i^t) \leq f(x_i^t) + \Delta_i^t \nabla f(x_i^t + \sigma_i^t \Delta_i^t d_i)^T d_i$$

for some $\sigma_i^t \in [0, 1]$. Therefore,

$$0 \leq \nabla f(x_i^t + \sigma_i^t \Delta_i^t d_i)^T d_i, \quad t \in \hat{\mathcal{B}}_i.$$

Taking the limits as $t \to \infty$, we get

$$0 \leq \nabla f(\hat{x})^T d_i \quad \text{for all } i \in \mathcal{P}. \tag{8.11}$$

Since the vectors in $\mathcal{D}$ are assumed to form a positive spanning set for $\mathbb{R}^n$, (8.11) implies that $\nabla f(\hat{x}) = 0$. ∎

9. Conclusions. When developing this analysis, we tried to keep the number of assumptions made to a minimum. Our first priority was to assure that under standard assumptions, the version of APPS that we had implemented could be shown to be globally convergent. That said, there are some further relaxations we could have made. For instance, in (3.2) we assumed, for convenience, that all processes started with the same initial iterate x^0 and the same initial value Δ^0 for the step-length control parameter. While we could relax (3.2), to do so would introduce a level of complication to the analysis that does not appear to add appreciably to the fundamental result.

An extension of more obvious practical import is to allow the set of search directions to change over time. In this paper, we assume that the set of search directions is fixed. Earlier pattern search results [10] make clear that this condition can be

relaxed to allow a more general notion of *exploratory moves*. Experience with sequential implementations of pattern search has demonstrated that there certainly can be algorithmic advantage to doing so. For instance, the exploratory moves enable more aggressive or speculative steps that may either accelerate the progress of the search or move the iterates away from a local minimizer, without compromising global convergence. In the parallel setting, one of the motivations for APPS was to devise algorithms that could recover from the failure of a process. Since all we require, in the end, is that (8.11) holds for enough vectors in $\mathcal{D}$ to form a positive basis for $\mathbb{R}^n$, we have some flexibility in both the implementation and the analysis. In particular, exploratory moves are included implicitly. The exploratory moves play an active role in the search only if they produce a success, but our analysis focuses on the contractions. As long as we can express any point produced by an exploratory move as in Theorem 5.2 (i.e., any success produced by an exploratory move lies on an appropriate lattice), the analysis accommodates this extension in a straightforward fashion.

Another possible extension to the analysis is to examine the robustness of the search in the presence of process failures either because the processor on which the process resides fails or because on that particular process the evaluation of the function f at a given x fails. In the current implementation of APPS, we ignore the failure of a process so long as the search directions contained on the active processes continue to form a positive spanning set. If we experience enough process failures that this condition no longer holds, we restart enough processes so that the condition is once again satisfied. If we assume a finite number of failures for evaluation at a given point—an extension to (3.5), our assumption that the maximum number of time steps for evaluating f at a given x is finite—then the modifications required to the analysis seem straightforward enough that we simply note them here.

A more ambitious option, along the lines of related ideas proposed in [11, 8, 4], would be to actually change the set of search directions during the course of the search, rather than working with some subset of a fixed set of directions chosen at the start of the search. To do so requires some modification of the mechanism used to control the length of the step. Our analysis relies on the algebraic structure of the iterates. This can be relaxed, either by requiring Δ to go to zero in the limit [11, 4] or by introducing a sufficient decrease condition to determine the success of a step [8], in lieu of the simple decrease conditions in (3.7) and (3.11) that we use here.

We close with the observation that we can reduce the general framework presented here to a special case that looks more like traditional (sequential) pattern search. (This is what motivated us to allow $0 \leq \gamma$ so that communication can be "instantaneous," as it would be in the sequential case.) The difference here is that we have introduced the bounds given in (3.10) for $t \in \mathcal{S}_i$. These bounds are necessary for our analysis (e.g., in the proofs of Lemma 7.7 and Corollary 7.9 or for the definition of $\underline{\kappa}(\Delta)$ in (7.3), which plays a role in the proofs of Lemma 7.6, Corollary 8.2, and Lemma 8.4). Prior definitions of pattern search did not require the enforcement of (3.10) since the synchronization of the updates to Δ suffices without the imposition of these bounds on updates made after a successful step.

Acknowledgments. We thank both referees for their thoughtful comments. Among their many useful suggestions, they recommended that we provide the summaries in what is now section 4 and in the opening of what is now section 8. The presentation has been much improved by their recommendations. We also thank Margaret Wright, the associate editor, for her helpful suggestions and for her handling of the manuscript.

REFERENCES

[1] C. Audet and J. E. Dennis, Jr., *Pattern search algorithms for mixed variable programming*, SIAM J. Optim., 11 (2000), pp. 573–594.

[2] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] J. Céa, *Optimisation: Théorie et algorithmes*, Dunod, Paris, 1971.

[4] I. D. Coope and C. J. Price, *On the convergence of grid-based methods for unconstrained optimization*, SIAM J. Optim., 11 (2001), pp. 859–869.

[5] P. D. Hough, T. G. Kolda, and V. J. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 134–156.

[6] T. G. Kolda and V. J. Torczon, *Understanding asynchronous parallel pattern search*, in High Performance Algorithms and Software for Nonlinear Optimization, G. DiPillo and A. Murli, eds., Kluwer Academic, Boston, 2003, pp. 316–335.

[7] R. M. Lewis and V. Torczon, *Rank Ordering and Positive Bases in Pattern Search Algorithms*, Tech. Rep. TR 96–71, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1996.

[8] S. Lucidi and M. Sciandrone, *On the global convergence of derivative-free methods for unconstrained optimization*, SIAM J. Optim., 13 (2002), pp. 97–116.

[9] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.

[10] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.

[11] W. Yu, *Positive basis and a class of direct search techniques*, Sci. Sinica, Special Issue I on Math., 1 (1979), pp. 53–67 (in Chinese).