# Multi-Agent Collaboration in Adversarial Environments*

K. Mani Chandy, Michel Charpentier† and Concetta Pilotto
Computer Science Department, California Institute of Technology, Mail Stop 256-80
Pasadena, California 91125
mani@cs.caltech.edu, charpov@cs.unh.edu, cetta@caltech.edu

## Abstract

*This paper presents a strategy for developing distributed algorithms for a class of problems by representing the problems as constrained optimizations and then employing a generic technique for deriving distributed optimization algorithms. In the distributed constrained optimization algorithm, multiple agents collaborate to optimize global, system-wide, objective functions while satisfying global constraints though each agent has only limited local information. The strategy can be employed in dynamic systems in which the numbers and locations of agents vary arbitrarily as computations proceed. This paper deals with problems in which agents operate in adversarial or uncertain environments and where the goal is to maximize the strength of the weakest link in a system.*

## 1 Introduction

**Agents and their Goals** A conventional model of distributed algorithms is a process network represented as a directed graph in which vertices represent processes and edges represent communication channels. Processes are described as message-passing programs with rules often of the form: When a message $m$ arrives, take a specified action $a$. While this model has demonstrated its utility over many years it is less suited for dynamic or adversarial environments in which communication structures change in unpredictable ways and in which agents can be added to or deleted from the system.

Consider agents entering a burning building or dealing with a hurricane. The ability of agents to communicate may vary from instant to instant, agents may die, and new agents may be added to deal with the situation. Examples from biology include that of ants building ant hills where the conformations of hills depend on the amounts of mud and grass in the vicinity. Ants may die, some born, ants wind-blown from one location to another, channels of communication altered, and resources (mud) washed away. Process network models are not ideal representations of these situations. Likewise, specifying precisely what each agent should do in each situation is unwieldy because the number of situations an agent may face is so large. Programs that specify the procedure to be executed by each process for each incoming message and each state are inappropriate for systems in which state changes continuously.

An alternate approach is to specify goals—objective functions—and constraints for agents. Designers don't specify precisely what each agent should do in each situation. Instead, what designers require is *i*) *safety*: agents don't take actions that reduce their objective values or that violate their constraints, and *ii*) *progress*: given enough opportunities for increasing their objective function values they will do so eventually. Designers' proof obligations are to show that no matter what situations agents may face, when each agent strives to maximize that agent's objective function the overall global goal is achieved.

The approach of designing distributed systems by assigning goals and constraints to agents has been studied extensively in control theory, game

*Submitted to the 21st Symposium on Distributed Computing (DISC'2007). This research is supported in part by AFOSR MURI award FA9550-06-1-0303.

†On leave from the University of New Hampshire, 2006/07

theory, and artificial intelligence. A contribution of this paper is to show how distributed computing concepts—particularly ideas of fairness, temporal logic and group communication—can be used to design systems operating in environments about which little is known. We believe that distributed computing ideas can add significant power to the growing confluence of optimization, control and game theories.

Many papers in game and control theory assume synchronous rounds in which all agents make moves concurrently or move in an orchestrated fashion. Other papers use probabilistic approaches in which the choices that agents make are independent random variables with specific distributions. Many control theory papers study systems in which dynamics are specified by differential equations showing relationships between agents specified by connectivity or communication graphs. This paper shows how distributed computing concepts can be used for dynamic systems in which: there is no persistent connectivity between agents; agents don't make choices in an orchestrated fashion; agents may change state continuously (as in systems modeled by differential equations) or discretely; and agents don't have access to independent random number generators.

Our strategy for developing distributed algorithms is to first represent a given problem as a constrained optimization—this step is often straightforward. The next step is to design agents, i.e., specify their goals and constraints. This step can be difficult because we must prove that agents carrying out local optimizations based on local information collectively solve the global optimization problem. We show, however, that there is a class of problems for which this step is trivial. In this class, all agents in all situations always solve optimization problems with the same structure. Agents adapt their objective functions and constraints to the information available to them. Because all groups of agents behave in the same way we call these algorithms *self-similar*.

**Model** Our goal is to demonstrate the value added by distributed computing concepts to control and game theory. Therefore, our model of distributed systems has to allow for states and transitions that are continuous as well as discrete. Furthermore, our model must be able to represent different situations in which agents interact. These situations range from firefighters talking to each other while putting out a forest fire to conventional message-passing algorithms employed by mobile robots exploring a contaminated region. We use group operations as the basis for communication in our model.

Groups of agents meet for periods of time and execute atomic operations across the group. An operation conducted by a group may change the states of all agents in the group. These state changes may be continuous or discrete, and the degree of change may depend on the length of time that the group meets. Designers cannot predict which groups will meet nor the length of times of the meetings. In the context of agents dealing with a crisis, such as a fire, a group consists of a set of agents who can see or communicate reliably with each other. As agents move and as the fire changes direction an agent in the group may no longer be able to see another, and thus groups change in unpredictable ways. Likewise, a group of proximate agents communicating by wireless may enjoy reliable broadcast for some period of time, and lose communication as agents move.

Operations across groups are more likely to abort as the sizes of groups increase and rates of failures of agents and communication links increase. When systems are static, atomic operations on large groups often result in more rapid convergence of algorithms. Atomic operations on smaller groups can result in greater efficiency in dynamic systems because the operations on small groups complete successfully more often than operations on large groups. This paper presents algorithms that employ atomic operations across groups where the sizes and memberships of the groups may vary as computation

proceeds. Operations are executed opportunistically on larger groups where possible and smaller ones where necessary.

A great deal of work has been done on group communication algorithms in which an agent is guaranteed that a message sent by the agent is received by all agents in a group or by none. We implement group operations on message-passing communication infrastructures by employing a group communication algorithm.

**Paper Contributions** Distributed computing concepts, such as fairness and temporal logic, provide the capability of specifying and reasoning about systems with very weak constraints. Game, control and optimization theory do not have such concepts. By using distributed computing ideas we can show that certain algorithms in the literature of control theory can be applied more widely than indicated in that literature. Environments that cannot be specified conveniently, using concepts from these theories, can be specified simply with fairness assumptions that are traditional in distributed computing.

For example, an immense amount or research has been done on consensus problems in which dynamics are specified by differential equations relating values of neighboring agents in a graph. See [10] for a recent survey. We show that these algorithms work for a broader class of environments in which agent state transitions may be discrete or continuous, and agent-connectivity graphs may be unknown and dynamic. There is no convenient way to describe these systems exclusively in terms of the concepts of control or game theory. The point of this paper is that distributed computing theory provides powerful tools to solve problems that have, traditionally, fallen outside its scope.

Though we plan to demonstrate the value of distributed computing concepts to optimization and control problems generally, in this paper we restrict attention to a particularly simple constrained optimization structure and show its applicability to a class of problems. Using a simple structure helps illustrate the validity of distributed computing concepts to optimization. Later work will deal with more general structures.

We present a collection of simple problems that are trivial if all the input is known in advance to all agents. The problems are nontrivial when agents don't know all the data at initiation of the algorithm, and agents carry out computations concurrently with the acquisition of additional data. Agents cannot wait to start computing until all the data is acquired because they don't know when data acquisition is complete. The collection of simple problems in this paper are chosen because they can be illustrated briefly. We don't discuss these examples in detail.

## 2 Max-Min Problems.

We are given a set of $N$ points indexed by $i$, $0 < i \leq N$. Associated with each $i$ is the amount $V_i$ of resources allocated to it and a function $f_i$, which maps $V_i$ to the reals. The quantity $f_i(V_i)$ is the utility of having $V_i$ resources at location $i$. Our problem is to maximize a system-wide utility $z$ subject to the constraints that resources are conserved and that the amount of resource at each point is nonnegative. Agents operating in adversarial environments maximize the strength (utility) of the weakest link. In this case $z$ is the minimum utility over all points, and the problem is defined by:

$$\left.\begin{array}{l} \textit{maximize } z = \min\limits_{0<i\leq N} f_i(V_i) \quad \textit{subject to} \\ \textbf{Conservation law: } \sum\limits_{0<i\leq N} V_i = C \\ \wedge\ \textbf{Nonnegativity: } \forall i : 0 < i \leq N : V_i \geq 0 \end{array}\right\} \quad (1)$$

where $V_i$ are the unknown (assumed to be scalars) and $C$ is the total amount of resource available. A "point" does not necessarily mean a location in a geometrical space but represents an index in some set. In the remainder of this section, we give several examples of problems that

can be represented as Max-Min optimizations.

**Computing Statistics.** Distributed computations of many statistics can be represented as Max-Min problems. Consider the example of computing a weighted mean. Agent $i$ has two real numbered values $V_i$ and $W_i$ where $W_i$ is a constant and $V_i$ is variable with initial value $V_i^{(0)}$. The problem specification is that eventually each $V_i$ becomes and remains $k.W_i.V_i^{(0)}$, where $k$ is a constant of proportionality, while the sum of all $V_i$ remains unchanged. We represent this problem as a Max-Min problem by defining utility functions as follows:

$$f_i(V_i) = \frac{rV_i}{W_i}$$

where $r$ is a positive constant. The proof that the optimum solution to the Max-Min problem with these utility functions returns the weighted mean is trivial.

**Mobile-Agent Formation.** A number of agents, indexed $i$ where $0 \leq i \leq N$ $(N > 0)$, are placed in a straight line with agent $i$ between agents $i-1$ and $i+1$, for $0 < i < N$. The positions of agents indexed 0 and $N$ are fixed; let these positions be 0 and $C$, respectively along the line. The variable $V_i$ of agent $i < N$, is the distance between agents $i$ and $i+1$. Anthropomorphically speaking, agent $i$ can see how far away agent $i+1$ is from itself, but can't see anything else. An agent $i$ does not have any information about its index $i$ or about the total number of agents in the system or about $C$.

Agents $i = 1, 2, \cdots, N-1$ are required to move along the line so that the $V_i$ satisfy some criterion such as $V_i = KV_{i-1}$ for some constant $K$. In this case, with $N = 3$, $C = 7$ and $K = 2$, the optimum values of $V_i$ for $j = 0, 1, 2$ are, respectively: $1, 2, 4$.

This problem is a Max-Min optimization in which the utility functions are:

$$f_i(V_i) = \frac{rV_i}{K^i}$$

where $r$ is a positive constant. For the numerical example of the previous paragraph, the optimum $z$ is $z = r$ since

$$z = r = \frac{rV_0}{2^0} = \frac{rV_1}{2^1} = \frac{rV_2}{2^2}$$

The optimization problem for a contiguously indexed group of agents $i = I, I+1, \cdots, P$ is obtained in the obvious way: this group doesn't know about the existence of agents outside the group so agents in the group carry out optimization steps as though they were the only agents in the system. Therefore, they assume that agents at the ends of the group—the agents indexed $I$ and $P$—don't move. The other agents in the group may move in an optimization step and therefore the variables of the group operation are $V_i$ for $I \leq i < P$. Any move that increases the value of the objective function $\min_{I \leq i < P} V_i$ is a valid step of the optimization algorithm.

**Building Structures.** Consider the problem of creating structures where the size and shape of the structure depends on the amount of resource available. Restrict attention to a single resource with which the structure is built. The structure is defined by the height at point $i$, which is amount $V_i$ of resource at this point.

As an example, consider $2N$ points on a line of length $2L$. Our goal is to build a pyramid-like structure. The more resources we have, the higher the pyramid. However, we want to avoid pyramids with slopes larger than 1 (45 degrees). The highest such pyramid has height $L$. If the total amount $C$ of resource allows for a higher structure, we want it to be shaped as a "house" (a 45 degree pyramid on top of a rectangular base). Fig. 1 shows the desired structure for different values of $C$.

To show that this problem can be implemented as a Max-Min problem, we focus on the left half of the pyramid (the other half is designed in a symmetric way). Assume there are $N$ points on the left half of the line and each point $i$ has a position $x_i$ on the line. Then, the value $V_i$ represents the height of the structure at location $x_i$.

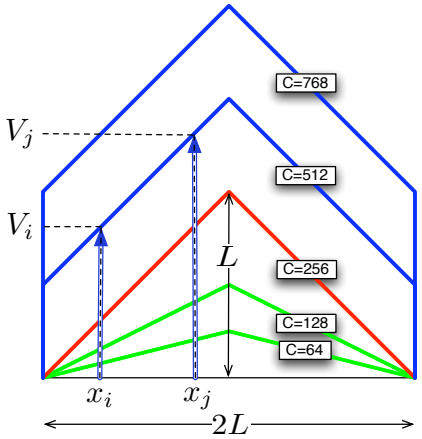


Figure 1: A pyramidal structure on a line

Assuming that $0 < x_i \leq L$ for all $i$, we consider the Max-Min problem defined by the following functions $f_i$:

$$f_i(V_i) = \begin{cases} rV_i/x_i & \text{if } V_i \leq x_i \\ r'(V_i - x_i) + r & \text{if } V_i > x_i \end{cases}$$

where $r$ and $r'$ are any positive constants. Because each function is strictly monotonic in the range $[0, C]$, it can be shown that the solution to this Max-Min problem is a consensus: $\forall i : z = f_i(V_i)$. If $C \leq \sum_i x_i$, this consensus is such that $z \leq r$. If $C$ is larger, the consensus $z$ becomes larger than $r$ (Fig. 2). In the first case ($z \leq r$), the optimum state satisfies $V_i = \frac{z}{r} x_i$ and the structure has a pyramid shape with slope $\frac{z}{r}$. In the second case ($z > r$), the optimum state satisfies $V_i = \frac{z-r}{r'} + x_i$ and the structure has a "house" shape with a 45 degree roof slope.

**Nondeterministic Structures.** This technique can be generalized to construct a variety of structures. If the functions $f_i$ are not strictly monotonic, the solution to a Max-Min problem may not be a consensus. In this case, the solution $z$ can be obtained through several possible vectors $(V_i)_{0<i\leq N}$. This means that the specification becomes nondeterministic and that several structures are acceptable. For instance, consider the following problem: we want $N$ agents on a

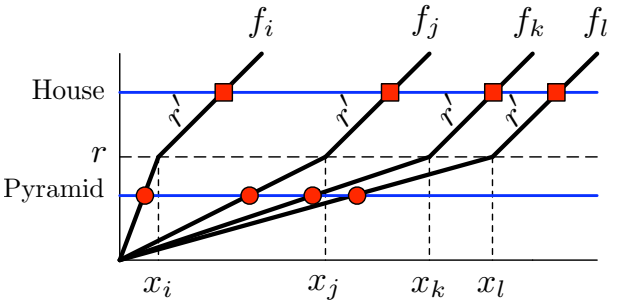


Figure 2: The pyramid/house problem as Max-Min

line to build a wall. Each agent $i$ has a location $x_i$ on the line. If there are enough resources $C$, we want this wall to be 10 meters high. If not, we want the wall to be as high as possible. Furthermore, we want the wall to be less that 20 meters high everywhere, if possible. Fig. 3 shows a possible solution for the case where $C$ is large enough to attain 10 meters, but not so large that the wall needs to be 20 meters anywhere.

Such a wall can be built by solving a Max-Min problem in which all the agents use the same utility function $f$ defined as:

$$f(V_i) = \begin{cases} rV_i & \text{if } V_i \leq 10 \\ 10r & \text{if } 10 < V_i < 20 \\ 10r + r'(20 - V_i) & \text{if } V_i \geq 20 \end{cases}$$

where $r$ and $r'$ are positive constants. Note that walls under 10 meters or above 20 meters will be flat. This is because all points are using the same function $f$ and these cases correspond to consensus solutions on the increasing or decreasing part of the curve. Non-consensus solutions correspond to walls like the one depicted in fig. 3.

## 3 Algorithm Development

**Self-Similar Strategy.** What should a group of agents do when they can execute atomic operations? The group has no information about the existence of other agents. So each group solves an optimization problem *pertinent to that group*. Each group determines an objective function and constraints pertinent to the group, and

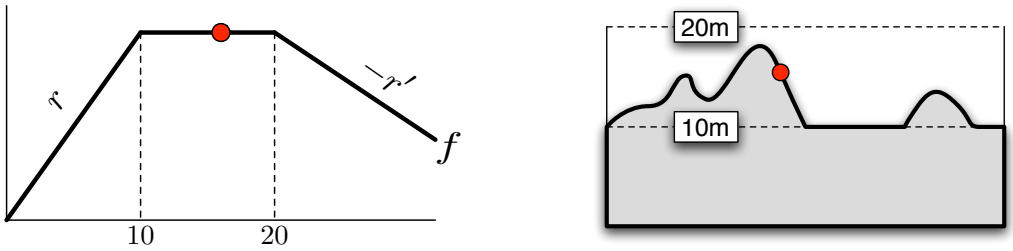


Figure 3: The wall problem as Max-Min and a possible solution

takes steps that ensure that group constraints remains satisfied while (preferably) improving the value of the objective function for the group or leaving the value unchanged. Informally speaking, each group of agents executes steps assuming that no other agents exist. We call this strategy "self-similar" because group operations are similar to subgroup operations. A proof structure for demonstrating the correctness of a self-similar strategy is to show the following:

**Safety** We show that any step taken by any group $G$ of agents that maintains (i.e., continues to satisfy) $G$'s constraints also maintains the global constraints.

**Termination** Many optimization algorithms are proved to converge only in the limit. Proofs of limits (for example showing that $e^{-t}$ tends to zero as $t$ tends to infinity) are inadequate for systems that involve discrete steps. We must prove that algorithms terminate and that when they do terminate the results are within specified tolerances of an optimum solution.

A computation in which there is *never* any interaction between agents in nonempty set $s$ and its complement $\bar{s}$ may be unable to achieve a global objective. Therefore, we assume the following environmental constraint on group formation: for every nonempty set $s$ of agents, an infinite number of groups are formed, which contain at least one agent from $s$ and one agent from $\bar{s}$, the complement of $s$.

This assumption can be formulated equivalently in terms of an undirected graph. Let $\mathcal{G}$ be a graph among agents defined as follows: $\mathcal{G}$ contains an edge $(i, j)$ if and only if an infinite number of groups are formed that contain both $i$ and $j$. The previous assumption is equivalent to saying that this graph $\mathcal{G}$ is connected. Designers do not have information about this graph. Furthermore, agents cannot learn about $\mathcal{G}$ because the graph is defined in terms of infinite computations. Therefore, agents must proceed opportunistically, using whatever groups are available to them.

**Self-Similar Max-Min Algorithms.** Consider steps of an algorithm taken by a group $G$ of agents to solve the problem pertinent to this group. A group $G$ takes a step in which each variable $V_i$ becomes $V_i'$ where:

$$\left.\begin{array}{c} \textbf{Changes in } G \textbf{ improve } G\textbf{'s minimum}: \\ (\forall i \in G : V_i'{=}V_i) \vee (\min\limits_{i\in G} f_i(V_i') > \min\limits_{i\in G} f_i(V_i)) \\ \textbf{Conservation law in } G: \\ \sum\limits_{i\in G} V_i' = \sum\limits_{i\in G} V_i \\ \textbf{Nonnegativity in } G: \\ \forall i \in G : V_i' \geq 0 \end{array}\right\} (2)$$

The behavior of groups is nondeterministic: any step that maintains the sum and increases the objective function is a valid step. If no such step is possible then the states of agents in the group are left unchanged.

The main result of the paper is the following theorem.

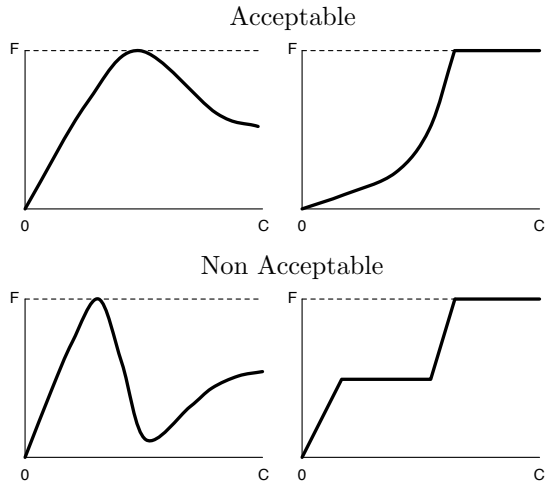


Figure 4: Sample acceptable and non acceptable functions $f_i$

**Theorem 1.** *The self-similar distributed strategy* (2) *solves the Max-Min optimization problem* (1) *if the utility functions $f_i$ satisfy the following properties, for all $i$:*

1. *$f_i$ is continuous in $[0, C]$;*
2. *$f_i(0) = 0$ and $\forall x \in [0, C] : f_i(x) \geq 0$;*
3. *A local maximum of $f_i$ in the range $[0, C]$ is also the global maximum in the range, denoted by $F_i$;*
4. *$f_i$ is quasiconcave.*

This theorem is a consequence of theorems 2, 3 and 4, which are stated in sect. 4.

**Quasiconcavity.** A function $f$ is quasiconcave if for any points $x$, $y$ in its domain, the value of the function at any point between $x$ and $y$ is greater than or equal to the smaller of $f(x)$ and $f(y)$:

$$\forall x, y \in [0, C] : \forall \lambda \in [0, 1] : \\ f(\lambda x + (1 - \lambda)y) \geq \min(f(x), f(y))$$

Quasiconcavity characterizes those functions that are first increasing, then (possibly) decreasing. Fig. 4 shows two examples of quasiconcave functions with no nonglobal maxima, as well as two examples that do not fit our assumption (the first is not quasiconcave and the second has a nonglobal maximum). Note that acceptable functions $f_i$ may not be monotonic, may reach their maximum on several points, and may not be differentiable everywhere. The most general shape of a quasiconcave function in which all local maxima are also global is represented in fig. 5: an initial strictly increasing part, followed by a "flat" part, followed by a strictly decreasing part.

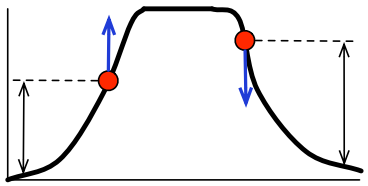

Figure 5: Quasiconcave with no nonglobal maximum

# 4 Correctness of Self-Similar Max-Min Algorithms

To verify that global constraints are satisfied after a step defined by (2) given that they are satisfied before the step is straightforward. Observe that a step by any group increases the value of the global objective function or leaves it unchanged. Therefore, we restrict attention to proving that the algorithm terminates, and that the values at termination are optimum. Proofs are given in appendix A.

**Algorithm Termination.** With a continuous state space, the algorithm may make ever smaller changes to the state and never terminate (though it may converge toward a useful limit). Instead of considering the general question of convergence, we instead discretize the state space in the following way. We assume a step value $\Delta$ and we modify the variables $V_i$ by steps that are multiple of $\Delta$. Therefore, the smallest change to a variable $V_i$ is to increase or decrease by $\Delta$. When

no such changes can improve the state, the algorithm terminates.

**Theorem 2.** *Given a choice of $\Delta > 0$, the algorithm terminates after a finite number of steps.*

**Properties of the System at Termination.** In this section, we state properties of the global state after the algorithm has terminated. In particular, we claim that, under the given assumption on group formation, the algorithm can be made to terminate in a state arbitrarily close to the true global optimum. More precisely, let $\epsilon$ be defined as follows:

**Definition 1.**

$$\epsilon = \max_i \max_{\substack{x,y\in[0,C] \\ |x-y|\leq\Delta}} |f_i(x) - f_i(y)|$$

Since the functions $f_i$ are assumed to be continuous, $\epsilon$ is a finite quantity and we have:

$$\lim_{\Delta\to 0} \epsilon = 0$$

Thus, $\epsilon$ can be made arbitrarily small by choosing $\Delta$ appropriately.

**Theorem 3.** *Let $\bar{z}$ be the exact solution to the given optimization problem and $z^*$ be the solution computed by the group-based algorithm upon termination (i.e., $z^*$ is the smallest $f_i(V_i)$ in the final state). Then:*

$$\bar{z} \geq z^* \geq \bar{z} - (\mathsf{diam}(\mathcal{G}) + 1)\epsilon$$

*where $\mathsf{diam}(\mathcal{G})$ is the diameter of the group formation graph $\mathcal{G}$ defined earlier.*

Note that the difference $\bar{z} - z^*$ can be made arbitrarily small by choosing an appropriate value of $\Delta$. In the worst case (linear graph $\mathcal{G}$), $\mathsf{diam}(\mathcal{G}) = N - 1$ and therefore the error $\bar{z} - z^*$ is always bounded by $N\epsilon$.

The proof of theorem 3 relies on a characterization of the state of groups after they terminate, that is when there is no $\Delta$ step that can improve the group state. This needs only to be done for groups of size 2, which is the worst case (larger groups can compute at least as good a solution as do groups of size 2).

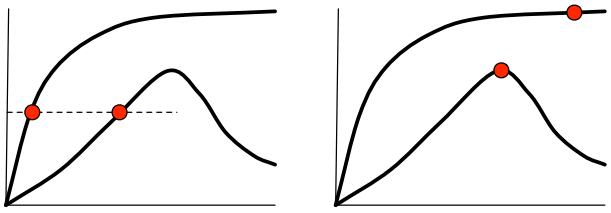

Figure 6: Consensus and non-consensus termination

Groups may terminate in a consensus (fig. 6, left part) or in a non-consensus state (fig. 6, right part). In the first case, it can be shown that $V_i$ and $V_j$ are both on the increasing part of their curve or both on the decreasing part, and that $f_i(V_i)$ and $f_j(V_j)$ are within $\epsilon$ of each other. In the second case, we show that one agent $i$ is within $\epsilon$ of the maximum value $F_i$ of its utility function and that the value of the other agent cannot be smaller.

Theorem 3 shows that the algorithm can be made to terminate in a state in which the smallest of the $f_i(V_i)$ is as close as desired from the true optimum $\bar{z}$. It remains to show that the algorithm can compute values of $V_i$ as close as desired to optimum values.

We define a family of values $\gamma_i$ that relate $\epsilon$ to the error on $V_i$:

**Definition 2.**

$$\gamma_i = \max_{\substack{x,y\in[0,C] \\ |f_i(x)-f_i(y)|\leq\epsilon \\ ((\mathsf{incr}_i(x)\wedge\mathsf{incr}_i(y)) \\ \vee(\mathsf{decr}_i(x)\wedge\mathsf{decr}_i(y)))}} |x - y|$$

Each $\gamma_i$ is to $\epsilon$ what $\epsilon$ is to $\Delta$. In particular, $\gamma_i$ enjoys a similar limit property, which follows from the continuity of the functions $f_i$:

$$\forall i : \lim_{\epsilon\to 0} \gamma_i = 0$$

Each $\gamma_i$ can be made arbitrarily small by choosing a suitable $\Delta$.

**Theorem 4.** *The solution to Max-Min computed by the group-based algorithm upon termination is such that:*

$$\forall i : |V_i - \bar{V}_i| \leq \gamma_i$$

*where* $(\bar{V}_i)_i$ *is some exact solution to the optimization problem.*

## 5 Experimental Results

We have conducted a set of experiments, both for the consensus and the non-consensus cases. Two questions were primarily explored: *i)* Should agents tend to form large groups or small groups? Intuition may suggest that algorithms that use large groups converge faster; however, an atomic group operation on a large group is more likely to be aborted than an operation on a small group because a component of the group gets disabled while the operation is in progress. *ii)* How does the connectivity of the graph impact convergence? Intuition suggests that algorithms on densely connected graphs will converge faster than algorithms on sparse graphs because more information is fused at each step in densely connected graphs.

In this section, we present results from a simple experiment on the impact of the probability of abortion of atomic group operations. This experiment was done on a consensus case (simple average).

The probability that an atomic operation on a group succeeds is a complex function of the size of the group, the connectivity of agents in the group, and the number of primitive operations (such as sending/receiving a single message, and addition or multiplication) required to complete the group operation. Here, we focus exclusively on group size; for these experiments we assume that abortion rates are independent of connectivity within a group and the number of primitive operations. Let $\delta$ be the probability that an agent in a group fails during a group operation. Then the probability that an operation on a group of size $n$ fails is $p(n) = 1-(1-\delta)^n$. Fig. 7

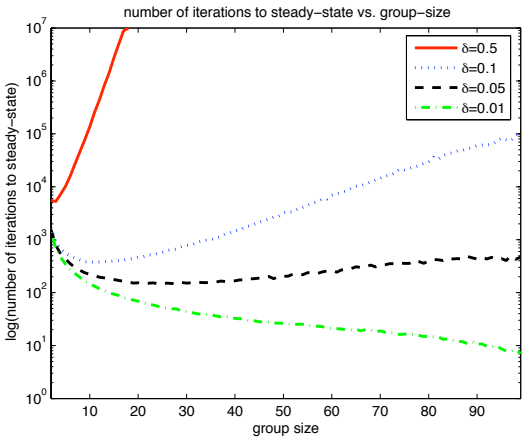


Figure 7: Impact of failure of group operations

shows how the number of time steps varies as a function of $\delta$ and group size for a 100-node network. As expected when $\delta$ is small (0.01) large group sizes result in fewer time steps. As $\delta$ increases to 0.05, a range of group sizes from about 10 to 50 have approximately minimum numbers of time steps, and when $\delta$ is large (0.5) any increase in group size is deleterious. These experiments support the intuition that even when the environment forms large groups, greater efficiency can be obtained by agents forming smaller subgroups if the probability of abortion of atomic operations increases with group size. Other experiments, not reported here, also validate our intuition about graph connectivity.

## 6 Related Work

This paper is based on a great deal of earlier work on group communication, consensus algorithms, optimization theory, and dynamic distributed systems; only a tiny part of this literature is referenced here. An architecture for survivable coordination of agents in systems in which some agents may behave arbitrarily is discussed in [6]. A great deal of work has been carried out on group communication upon which this paper is based [4, 9, 14, 3]. Decentralized iterative schemes for consensus problems have

been proposed by [16]. Consensus problems, particularly computations of averages in dynamic networks of agents are treated in [11, 15, 8]. General references on convex optimization and quasiconcavity are [12, 5]. Distributed algorithms, some based on convex optimization, are presented in [1, 7, 13]. Convergence in multi-agent coordination has been discussed in several papers such as [2].

## References


[1] D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.

[2] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis. Convergence in multiagent coordination consensus and flocking. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2996–3000, 2005.

[3] B. Charron-Bost. Agreement problems in fault-tolerant distributed computing. In *28th Annual Conference on Current Trends in Theory and Practice of Informatics*, volume 2234 of *LNCS*, pages 10–32, 2001.

[4] N. Lynch. *Distributed algorithms*. Morgan Kaufmann Publishers, 1997.

[5] S. S. M. Avriel, W. E. Diewert and I.Zang. *Generalized Concavity*. Plenum Press, 1988.

[6] D. Malkhi and M. K. Reiter. An architecture for survivable coordination in large distributed systems. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):187–202, 2000.

[7] P. Ogren, E. Fiorelli, and N. Leonard. Cooperative control of mobile sensor networks: adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, 2004.

[8] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sept. 2004.

[9] A. S. R. Guerraoui. Genuine atomic multicast in asynchronous distributed systems. *Theoretical Computer Science*, 254(1,2):297–316, 2001.

[10] A. F. R. Olfati-Saber and R. Murray. Consensus and cooperation in networked multiagent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[11] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multiagent coordination. In *Proceedings of the 2005 American Control Conference*, pages 1859–1864, 2005.

[12] L. V. S. Boyd. *Convex Optimization*. Cambridge University Press, 2004.

[13] D. Scherber and H. Papadopoulos. Locally constructed algorithms for distributed computations in ad hoc networks. In *Proceedings of the third international symposium on Information Processing on Sensor Networks*, pages 11–19, 2004.

[14] A. Schiper. Dynamic group communication. *Distributed Computing*, 18(5):359–374, 2006.

[15] D. Spanos, R. Olfati-Saber, and R. Murray. Dynamic consensus on mobile networks. In *Proceedings of the sixteenth IFAC world congress*, 2005.

[16] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stocastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, Sept. 1986.

# A Proofs

## A.1 Quasiconcavity

Quasiconcavity characterizes those functions that are first increasing, then (possibly) decreasing. We define "increasing" and "decreasing" formally as follows:

**Definition 3.** We define the predicate $\mathsf{incr}_i(x)$ (resp. $\mathsf{decr}_i(x)$) to express that function $f_i$ is locally increasing (resp. decreasing) at point $x$:

$$\mathsf{incr}_i(x) \equiv \exists r > 0 : \forall d : 0 < d < r : f_i(x-d) < f_i(x) < f_i(x+d)$$
$$\mathsf{decr}_i(x) \equiv \exists r > 0 : \forall d : 0 < d < r : f_i(x-d) > f_i(x) > f_i(x+d)$$

Quasiconcave functions with no nonglobal maxima satisfy the following lemma:

**Lemma 1.** *For any function $f_i$ subject to our assumptions and any point $x$ in $[0, C]$:*

$$\mathsf{incr}_i(x) \implies (\forall y : y < x : (f_i(y) < f_i(x)) \wedge \mathsf{incr}_i(y))$$
$$\mathsf{decr}_i(x) \implies (\forall y : y > x : (f_i(y) < f_i(x)) \wedge \mathsf{decr}_i(y))$$

*Proof.* This lemma is a consequence of the general shape of quasiconcave functions with no nonglobal maximum (fig. 5). Each function consists of an initial increasing part[1], possibly followed by an optional "flat" part (with may be reduced to a single point), possibly followed by an optional decreasing part. One can see that a flat part cannot be followed by an increasing part (from the assumption that local optima are global optima) and a decreasing part cannot be followed by a flat part (again from the assumption that local optima are global optima) or by an increasing part (from the assumption of quasiconcavity). From this, it follows that $\mathsf{incr}_i(x)$ implies that $x$ is on the left part of the curve and everything left of $x$ is lower and also increasing. Similarly, $\mathsf{decr}_i(x)$ places $x$ on the right part of the curve (if any) and everything right of $x$ is lower and also decreasing. □

## A.2 Termination

A group may increase its minimum without increasing the minimum of the entire system. Therefore, we seek a variant function $h$ such that any change by any group changes the value of $h$ globally as follows:

$$h(G) = \mathsf{sort}(f_i(V_i))_{i \in G}$$
$$\prec = \text{lexicographic ordering}$$

where $\mathsf{sort}$ is the function that sorts a vector in nondecreasing order.

**Group progress.** It is clear that a group step from $G$ to $G'$ increases $h$ for this group w.r.t. lexicographic ordering ($h(G') \succ h(G)$). This is because the first component of vector $h$ for the group is the minimum of $f_i(V_i)$ for this group and this minimum increases.

[1] This part is empty for the special case of the flat constant function zero.

**Global progress.** Moreover, if several disjoint groups all increase their $h$, the smallest value that changes in the system is the minimum of some group, which is increasing. Therefore, the global $h$ also increases, according to lexicographic ordering. It follows that every effective group step improves function $h$ for the global state.

*Proof of theorem 2.* Let $F$ be a sorted vector of $F_i$ and $\leq$ applied to vectors be the element-by-element comparison. It is straightforward to show that the algorithm satisfies the following invariants:

$$\textbf{invariant } \forall i : 0 \leq V_i \leq C \tag{3}$$
$$\textbf{invariant } \forall i : f_i(V_i) \leq F_i \tag{4}$$
$$\textbf{invariant } h(S) \leq F \tag{5}$$

Invariant (5) follows from invariant (4), which follows from invariant (3), which follows from the conservation law and the nonnegativity assumption on group steps.

Since the values $V_i$ are only increased or decreased by multiples of $\Delta$, invariant (3) implies that each $V_i$ can only take a finite number of different values. Therefore, $h(S)$ can take only a finite set of values as well. From invariant (5), function $h$ is bounded above. As a consequence, the range of $h$ is well founded for the lexicographic order $\succ$. Since each state change increases $h$ the algorithm eventually terminates. □

## A.3 Properties at Termination

**Lemma 2.** *Given a choice of smallest step $\Delta$, a group $\{i, j\}$, upon termination, satisfies the following predicate:*[2]

$$\left.\begin{array}{l} \lor\ F_i - \epsilon \leq f_i(V_i) \leq f_j(V_j) \\ \lor\ F_j - \epsilon \leq f_j(V_j) \leq f_i(V_i) \end{array}\right\} \begin{array}{c} non \\ consensus \end{array}$$

$$\left.\begin{array}{l} \lor\ \land\ |f_i(V_i) - f_j(V_j)| \leq \epsilon \\ \quad \land\ \lor\ \mathsf{incr}_i(V_i) \land \mathsf{incr}_j(V_j) \\ \quad\quad \lor\ \mathsf{decr}_i(V_i) \land \mathsf{decr}_j(V_j) \end{array}\right\} consensus \tag{6}$$

*where $\epsilon$ is defined from $\Delta$ by def. 1. (Recall that $F_i$ is the maximum of function $f_i$ over the range $[0, C]$.)*

*Proof.* When the group $\{i, j\}$ terminates, there is no possible improvement of the minimum of the group by a $\Delta$ step. Given that the sum $V_i + V_j$ remains constant, there are two possible $\Delta$ steps

[2] When dealing with long logical formulas, we use the aligned list syntax advocated in [17]. Using this syntax, the formula $A \lor B \lor (C \land (D \lor E))$ is represented as:

$$\begin{array}{l} \lor\ A \\ \lor\ B \\ \lor\ \land\ C \\ \quad \land\ \lor\ D \\ \quad\quad \lor\ E \end{array}$$

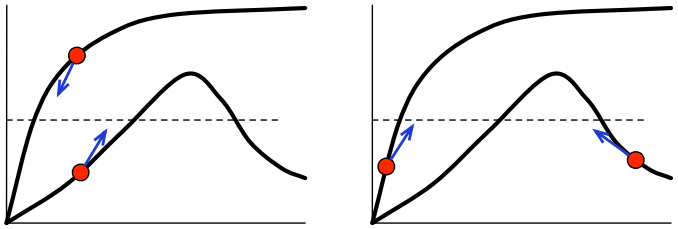

Figure 8: Possible steps (consensus case)

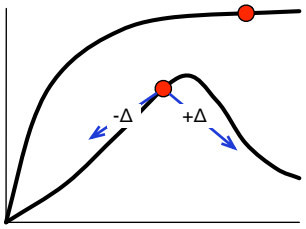


Figure 9: Possible steps (non-consensus case)

for the group:

$$V_i := V_i + \Delta, \qquad V_j := V_j - \Delta, \text{ or} \qquad \text{(Step A)}$$
$$V_i := V_i - \Delta, \qquad V_j := V_j + \Delta \qquad \text{(Step B)}$$

The algorithm stops for the group when each of these steps is either impossible or is not an improvement.

**Impossible steps.** Step A may be impossible because $V_i < \Delta$ or $V_j > C - \Delta$. But $V_j > C - \Delta$ implies $V_i < \Delta$ because $V_i + V_j \leq C$. Therefore, $V_i < \Delta$ when Step A is not possible. Similarly, $V_j < \Delta$ when Step B is not possible.

**Non-improvement steps.** If Step A is possible, it may still not be an improvement for the group $\{i, j\}$. This happens when the minimum of the group after the step is not larger than what it was before the step:

$$\min(f_i(V_i + \Delta), f_j(V_j - \Delta)) \leq \min(f_i(V_i), f_j(V_j))$$

A similar formula characterizes the case where Step B is possible but is not an improvement. From these conditions on impossibility and non-improvement, several cases need to be considered to derive formula (6).

**Non-improvement: consensus case.** Fig. 8 shows the intuition behind the consensus case (third disjunct). If $f_i(V_i)$ and $f_j(V_j)$ are more than $\epsilon$ apart (Fig. 8, left diagram), there is a possible $\Delta$ step in which the low point increases while the other does not decrease below the minimum as it was before the step. Furthermore, if one point is on the increasing part of its curve and the other on the decreasing part of its curve, there is a possible $\Delta$ step in which *both* utility functions will increase, even if the points are less that $\epsilon$ apart (Fig. 8, right diagram).

**Non-improvement: non-consensus case.** The idea behind the non-consensus case is illustrated in fig. 9. If the low point $f_i(V_i)$ cannot be moved because neither a $+\Delta$ step nor a $-\Delta$ step is an improvement, it follows that the utility function $f_i$ reaches a local maximum between $V_i - \Delta$ and $V_i + \Delta$. By assumption, this local maximum must be the global maximum $F_i$, and $f_i(V_i)$ cannot be more than $\epsilon$ away from it.

**Impossible steps.** Cases in which a point is prevented from moving because of the lower bound 0 are treated in a similar fashion (see appendix A.4 for details). □

The next stage in the proof of theorem 3 is to show that if one agent does not satisfy the desired bound on the error in the final state, then all agents are away from the optimum by at least $\epsilon$.

**Lemma 3.** *Given a choice of smallest step $\Delta$, the entire set of agents, upon termination, satisfies the following predicate:*

$$(\exists \alpha : f_\alpha(V_\alpha) < \bar{z} - (\mathsf{diam}(\mathcal{G}) + 1)\epsilon) \implies (\forall i : f_i(V_i) < \bar{z} - \epsilon)$$

*where $\bar{z}$ is the true optimal solution to the problem.*

*Proof.* Let $D = \mathsf{diam}(\mathcal{G})$ and $\alpha$ be such that $f_\alpha(V_\alpha) < \bar{z} - (D+1)\epsilon$. Let $r$ be the shortest distance between $\alpha$ and some agent $i$ in graph $\mathcal{G}$. Then the lemma follows from $r \leq D$ and:

$$f_i(V_i) < \bar{z} - (D + 1 - r)\epsilon \tag{7}$$

The proof of (7) is by induction. The base case ($r = 0$, $i = \alpha$) is clear. The induction step assumes (7) for an agent $i$ at distance $r$ from $\alpha$ and the proof obligation is that a neighbor $j$ of $i$, at distance $r + 1$ from $\alpha$, satisfies:

$$f_j(V_j) < \bar{z} - (D + 1 - (r + 1))\epsilon \tag{8}$$

By the definition of graph $\mathcal{G}$, a group is formed infinitely often that contains both agents $i$ and $j$. Because we assume the algorithm has terminated, this group cannot take any $\Delta$ step to improve its state. Consequently, the group $\{i, j\}$ cannot take a $\Delta$ improvement step either and hence satisfies condition (6) from lemma 2. By weakening, this condition implies that the group $\{i, j\}$ then satisfies:

$$\vee\ F_i - \epsilon \leq f_i(V_i) \leq f_j(V_j) \tag{9a}$$
$$\vee\ F_j - \epsilon \leq f_j(V_j) \leq f_i(V_i) \tag{9b}$$
$$\vee\ |f_i(V_i) - f_j(V_j)| \leq \epsilon \tag{9c}$$

**Consensus termination.** If group $\{i, j\}$ terminates in a consensus state (disjunct (9c)), agents $i$ and $j$ are within $\epsilon$ of each other and (8) follows from (7) easily.

**Non-consensus termination: agent $j$ at its maximum.** If agent $j$ is within $\epsilon$ of its maximum, disjunct (9b) implies that:

$$f_j(V_j) \leq f_i(V_i) < \bar{z} - (D + 1 - r)\epsilon < \bar{z} - (D + 1 - (r + 1))\epsilon$$

**Non-consensus termination: agent $i$ at its maximum.** This scenario is impossible. Using the fact that $\bar{z} \leq F_i$ and $r \leq D$, it follows from the assumption (7) on $i$ that:

$$f_i(V_i) < \bar{z} - (D + 1 - r)\epsilon \leq F_i - (D + 1 - r)\epsilon \leq F_i - \epsilon$$

This implies that $F_i - f_i(V_i) > \epsilon$, which contradicts disjunct (9a).

This completes the proof of (7). For any agent $i$, the distance $r$ to $\alpha$ is at most $D$, from which the lemma to be proved follows. □

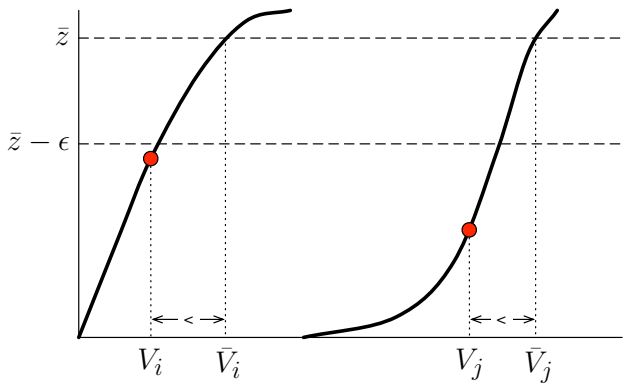


Figure 10: Case $\forall i : \mathsf{incr}_i(V_i)$ and $\sum_i V_i < C$

*Proof of theorem 3.* The part $\bar{z} \geq z^*$ is trivial from the fact that $\sum_i V_i$ is conserved and the $V_i$ remain nonnegative: Under this constraint, no computed solution can be better than the true optimum. The second conjunct is proved by contradiction. If it is assumed to be false, lemma 3 implies that $\forall i : f_i(V_i) < \bar{z} - \epsilon$.

Let $(\bar{V}_i)_{0<i\leq N}$ be any assignment to variables $V_i$ that is solution to the optimization problem:

$$\begin{aligned} &\wedge \min_{0<i\leq N} f_i(\bar{V}_i) = \bar{z} \\ &\wedge \sum_{0<i\leq N} \bar{V}_i = C \\ &\wedge \forall i : 0 < i \leq N : V_i \geq 0 \end{aligned}$$

Then, for all $i$:

$$f_i(V_i) < \bar{z} - \epsilon < \bar{z} \leq f_i(\bar{V}_i) \tag{10}$$

**Case $\forall i : \mathsf{incr}_i(V_i)$ (fig. 10).** If $\bar{V}_i \leq V_i$, then $f_i(\bar{V}_i) \leq f_i(V_i)$ because $\mathsf{incr}_i(V_i)$, using lemma 1. This contradicts (10) and therefore, $\bar{V}_i < V_i$, for all $i$. Hence, it follows that $\sum_i V_i < \sum_i \bar{V}_i = C$, which is impossible since, in any reachable state of the algorithm, $\sum_i V_i = C$. The case $\forall i : \mathsf{decr}_i(V_i)$ is similar.

**Case $\mathsf{incr}_i(V_I)$ and $\mathsf{decr}_j(V_j)$.** Since the graph $\mathcal{G}$ is connected, we can choose $i$ and $j$ to be neighbors in $\mathcal{G}$. Therefore, there is a group, containing both agents $i$ and $j$, that has reached its termination condition. It follows that the group $\{i, j\}$ itself cannot improve its state and satisfies condition (6) from lemma 2. Of the three disjuncts of (6), The first disjunct implies $f_i(V_i) \geq F_i - \epsilon \geq \bar{z} - \epsilon$, which contradicts the assumption $f_i(V_i) < \bar{z} - \epsilon$. The second disjunct leads to a similar contradiction with agent $j$. The third disjunct directly contradicts the case assumption that $\mathsf{incr}_i(V_i)$ and $\mathsf{decr}_j(V_j)$. □

## A.4 Detailed Proof of Lemma 2

We prove in detail a weaker form of the lemma, namely:

$$\begin{aligned}
&\vee\ (f_i(V_i) \leq f_j(V_j)) \wedge (F_i - f_i(V_i) \leq \epsilon) \\
&\vee\ (f_j(V_j) \leq f_i(V_i)) \wedge (F_j - f_j(V_j) \leq \epsilon) \\
&\vee\ |f_i(V_i) - f_j(V_j)| \leq \epsilon
\end{aligned}$$

Similar arguments can be used to prove (6).

*Proof.* The group $\{i, j\}$ terminates when there is no possible improvement of the minimum of the group by a $\Delta$ step. Given that the sum $V_i + V_j$ remains constant, there are two possible $\Delta$ steps for the group: $V_i := V_i + \Delta, V_j := V_j - \Delta$ or $V_i := V_i - \Delta, V_j := V_j + \Delta$. The algorithm stops for the group when each of these steps is either impossible or is not an improvement.

A step may be impossible because $V_i$ or $V_j$ is withing $\Delta$ of a bound of the range $[0, C]$. If $V_i$ is within $\Delta$ of $C$, then $V_i$ is within $\Delta$ of 0 because $V - i + V_j \leq C$ (and vice-versa). Therefore, we only need to consider impossibility from $V_i$ or $V_j$ being smaller than $\Delta$. The condition for group termination can be written as: The first step is impossible or is not and improvement, and similarly for the second step:

$$\begin{aligned}
&\wedge \vee V_j < \Delta \\
&\quad \vee \min(f_i(V_i + \Delta), f_j(V_j - \Delta)) \leq \min(f_i(V_i), f_j(V_j)) \\
&\wedge \vee V_i < \Delta \\
&\quad \vee \min(f_i(V_i - \Delta), f_j(V_j + \Delta)) \leq \min(f_i(V_i), f_j(V_j))
\end{aligned}$$

**Case 1:** Assume $f_i(V_i) \leq f_j(V_j)$, hence $\min(f_i(V_i), f_j(V_j)) = f_i(V_i)$. Using $\min(x, y) \leq a \equiv (x \leq a) \vee (y \leq a)$ and distributing $\wedge$ over $\vee$, the termination condition becomes:

$$\begin{aligned}
&\vee\ (f_i(V_i + \Delta) \leq f_i(V_i)) \wedge (f_i(V_i - \Delta) \leq f_i(V_i)) && \text{(a)} \\
&\vee\ (f_i(V_i + \Delta) \leq f_i(V_i)) \wedge (f_j(V_j + \Delta) \leq f_i(V_i)) && \text{(b)} \\
&\vee\ (f_i(V_i - \Delta) \leq f_i(V_i)) \wedge (f_j(V_j - \Delta) \leq f_i(V_i)) && \text{(c)} \\
&\vee\ (f_j(V_j - \Delta) \leq f_i(V_i)) \wedge (f_j(V_j + \Delta) \leq f_i(V_i)) && \text{(d)} \\
&\vee\ (f_i(V_i + \Delta) \leq f_i(V_i)) \wedge (V_i < \Delta) && \text{(e)} \\
&\vee\ (f_j(V_j - \Delta) \leq f_i(V_i)) \wedge (V_i < \Delta) && \text{(f)} \\
&\vee\ (f_i(V_i - \Delta) \leq f_i(V_i)) \wedge (V_j < \Delta) && \text{(g)} \\
&\vee\ (f_j(V_j + \Delta) \leq f_i(V_i)) \wedge (V_j < \Delta) && \text{(h)} \\
&\vee\ (V_i < \Delta) \wedge (V_j < \Delta) && \text{(i)}
\end{aligned}$$

Consider disjunct (a) and let $m$ be the maximum of $f_i$ over the range $[V_i - \Delta, V_i + \Delta]$. If $m = f_i(V_i - \Delta)$, then $m = f_i(V_i)$ because $f_i(V_i - \Delta) \leq f_i(V_i)$. Similarly, if $m = f_i(V_i + \Delta)$, then $m = f_i(V_i)$ because $f_i(V_i + \Delta) \leq f_i(V_i)$. Therefore, $\exists x \in ]V_i - \Delta, V_i + \Delta[ : m = f_i(x)$. Because $]V_i - \Delta, V_i + \Delta[$ is open, this $x$ is a local maximum of $f_i$ and, from the assumption that local maxima of $f_i$ are global maxima, $f_i(x) = F_i$. Because $x \in ]V_i - \Delta, V_i + \Delta[$, we know that $|f_i(x) - f_i(V_i)| \leq \epsilon$. Therefore, disjunct (a) implies $F_i - f_i(V_i) \leq \epsilon$.

The case of disjunct (e) is similar. Let $m$ be the maximum of $f_i$ over the range $[0, V_i + \Delta]$. If $m = 0$, then the function $f_i$ is constant and equal to 0 and $f_i(V_i) = F_i$. If $m > 0$, then $f_i(0) \neq m$ because $f_i(0) = 0$. If $f_i(V_i + \Delta) = m$, then $f_i(V_i) = m$ because $f_i(V_i + \Delta) \leq f_i(V_i)$. Therefore, in any case, $\exists x \in ]0, V_i + \Delta[ : m = f_i(x)$. The remainder of the proof for disjunct (e) is identical to the proof above, and we conclude $F_i - f_i(V_i) \leq \epsilon$.

For the five disjuncts (b), (c), (d), (f) and (h), the part $f_j(V_j \pm \Delta) \leq f_i(V_i)$ and the case assumption $f_i(V_i) \leq f_j(V_j)$ together imply that $f_i(V_i) \leq f_j(V_j) \leq f_i(V_i) + \epsilon$, hence that $|f_i(V_i) - f_j(V_j)| \leq \epsilon$.

The two remaining disjuncts, (g) and (i), include the part $V_j < \Delta$, from which we deduce, using the assumption $f_i(V_i) \leq f_j(V_j)$, that $0 \leq f_i(V_i) \leq f_j(V_j) \leq \epsilon$. It follows that $|f_i(V_i) - f_j(V_j)| \leq \epsilon$. Therefore, when all possible disjuncts are considered, the termination condition in case 1 implies:

$$((f_i(V_i) \leq f_j(V_j)) \wedge (F_i - f_i(V_i) \leq \epsilon)) \vee (|f_i(V_i) - f_j(V_j)| \leq \epsilon)$$

**Case 2:** Assume $f_i(V_i) \leq f_j(V_j)$. Using the same argument as in case 1, the termination condition can be shown to imply:

$$((f_j(V_j) \leq f_i(V_i)) \wedge (F_j - f_j(V_j) \leq \epsilon)) \vee (|f_i(V_i) - f_j(V_j)| \leq \epsilon)$$

Cases 1 and 2 together complete the proof. □

## References

[17] L. Lamport. How to write a long formula. *Formal Aspects of Computing*, 6(5):580–584, 1994.