

# AN ALGORITHM FOR GROUP FORMATION IN AN AMORPHOUS COMPUTER

RADHIKA NAGPAL \*

radhi@martigny.ai.mit.edu

MIT Artificial Intelligence Laboratory  
Cambridge MA 02139, U.S.A

DANIEL COORE

newts@martigny.ai.mit.edu

MIT Artificial Intelligence Laboratory  
Cambridge MA 02139, U.S.A

## Abstract

Amorphous computing[1] is the study of programming ultra-scale computing environments of smart sensors and actuators that communicate locally via wireless broadcast. In such environments, where individual elements have limited resources, aggregation into groups is useful for specialization, fault-tolerance, and resource allocation. This paper presents a new algorithm, called *clubs*, that takes advantage of the local communication to efficiently aggregate processors into groups in an amorphous computer. Time taken is proportional to the local density of processors, even in an asynchronous setting. The physical embedding of the amorphous computer is used to derive an upper bound on the number and density of groups formed. The clubs algorithm can be extended to adapt to processor failures and to find the maximal independent set (MIS) and  $\Delta + 1$  vertex coloring in  $O(\log N)$  rounds, where  $N$  is the total number of elements and  $\Delta$  is the maximum degree. Simulation results and example applications are presented.

**Keywords:** Parallel and Distributed Computing, Smart Sensors, Vertex Coloring, Maximal Independent Set, Symmetry Breaking.

## 1 Introduction

Recent developments in micro-fabrication and nanotechnology will enable the inexpensive manufacturing of massive numbers of tiny computing elements with integrated sensors and actuators. These smart agents can be embedded in structures to create active surfaces, improved materials and responsive environments [2, 3]. Amorphous computing [1] is the study of such ultra-scale computing environments. Aggregating processors into groups is a useful paradigm for increasing robustness, task specialization and efficient resource allocation [4, 5, 6]. This paper presents a new algorithm for forming groups efficiently in the context of an amorphous computer.

---

\*Nagpal and Coore are supported in part by DARPA under contract number N00014-96-1-1228 administered by ONR. Nagpal is also supported by Lucent GRPW Fellowship.

## 2 Computational Model

An amorphous computer consists of myriad identical processing elements. Each processor has limited computing resources and no globally unique identifier. Instead it has a random number generator for breaking symmetry. Processors have similar clock speeds but do not operate in lockstep. They are unreliable and may stop executing at any time. The processors have no precise interconnect. They are randomly and densely distributed on a surface or in a volume. These features make it possible to cheaply manufacture and program large quantities of smart elements, and embed them in materials. What makes an amorphous computer different from traditional distributed and parallel computers is the physical embedding of the amorphous computer and the communications model.

- Each processor communicates locally with processors within a circular region of radius  $r$ . The average local neighborhood size,  $d_{avg}$ , is much smaller than the total number of processors,  $N$ .
- Processors communicate with their local neighborhood by wireless broadcast. All processors share the same channel. Collisions occur when two or more processors with overlapping broadcast regions send messages simultaneously. Collisions result in those messages being lost. A processor listening to the channel can detect a collision because it receives a garbled message. However the sender cannot detect collisions because it cannot listen and transmit at the same time. This model is similar to that of multi-hop broadcast networks such as packet radio networks [7].

While this communication model allows for the simple assembly of large numbers of processors, the interference due to overlapping broadcast regions, lack of collision detection and processor asynchronicity make it difficult and inefficient to emulate reliable point-to-point communication. Group forming algorithms such as [5, 8], that are designed for point-to-point networks, are difficult to extend to the amorphous environment without a huge loss in efficiency. In addition, such al-

```

integer R (upper bound for random numbers)
boolean leader, follower = false

procedure CLUBS ()
1  ti := R
2  ri := random[0,R)
3  while (not follower and not leader)
4    ti = ti - 1
5    if (ri > 0)
6      ri := ri - 1
7      if (not_empty(msg_queue))
8        if (first(msg_queue) = "recruit")
9          follower := true
10     else
11       leader := true
12       broadcast("recruit")
13   while (ti ≥ 0)
14     listen for other leaders
15     ti = ti - 1

```

Figure 1: CLUBS ALGORITHM

gorithms often require synchronizers to function correctly in asynchronous environments [9]. These generate large numbers of messages, further exacerbating the problem of message loss. The clubs algorithm takes advantage of the local broadcast nature of the communications model and asynchronicity of the processors, to form groups efficiently and reliably.

### 3 Clubs Algorithm

In order for the groups to be useful for resource allocation and self-organizing communication networks, there are three requirements [6]. First, all processors must belong to some group. Second, all groups should have the same diameter. Third, a group should have local routing [4], which means that all processors within the group should be able to talk to each other using only processors within that same group. The clubs algorithm forms overlapping groups, called *clubs*, with a maximum group diameter of two hops.

Figure 1 presents the code run on a single processor. The processors start competing to form new groups by choosing random numbers from a fixed integer range  $[0, R)$ . Each processor counts down from that number silently. If it reaches zero without being interrupted, the processor becomes a group leader and recruits its local neighborhood into its group by broadcasting a “recruit” message. The processors that get recruited are called followers. Once a processor has been recruited as a follower, it stops counting down and listens for additional recruit messages. Groups are allowed to overlap. If a processor detects a collision (hears a garbled message) while counting down, it assumes that more than one of its neighbors tried to recruit it at the same time and becomes a follower. At the end

of  $R$  steps, all processors are leaders or followers, and the clubs formed satisfy the group requirements. We first present the analysis for synchronous processors and then extend it to the asynchronous case.

**Theorem 1:** *The clubs algorithm completes in  $R$  steps and produces valid groups, for a synchronous amorphous computer. (Proof omitted)*

The upper bound on the range of random numbers,  $R$ , is chosen so as to minimize the number of *leadership conflicts*. These occur when two neighboring processors declare leadership at the same time. Except when leadership conflicts occur, leaders are non-adjacent and at least distance  $r$  apart. For many applications of clubs it is desirable that group leaders belong to only one club (their own) and that the overlap between clubs be limited [6]. In addition the spacing between clubs allows us to place an upper bound on the number of groups formed (Section 3.1). Therefore, we would like to keep the number of leadership conflicts low.

**Theorem 2:** *The expected number of leadership conflicts,  $E(\text{conflicts})$ , is at most  $(\frac{d_{avg}}{2R})N$ , for a synchronous amorphous computer.*

**Corollary:** *If we choose  $R = \alpha d_{avg}$ , where  $\alpha$  is a constant and  $\alpha > 1$ , then the expected number of conflicts is at worst a constant fraction of the total number of nodes,  $(1/2\alpha)N$ .*

Proof omitted. Detailed proofs for all theorems in this paper are provided in [10]. Thus,  $\alpha$  can be chosen to make the percentage of leadership conflicts acceptably (or arbitrarily) small. The running time is  $O(d_{avg})$ , which is significantly smaller than  $N$ .

**Theorem 3:** *For asynchronous processors with the same clock speeds, the clubs algorithm completes in  $D + R$  steps, where  $D$  is the delay between when the first processor starts counting down and the last processor starts counting down. For processors with different clock speeds, the clubs algorithm completes in the time taken for the slowest processor to count  $R$  steps. The expected number of leadership conflicts  $E(\text{conflicts})$  remains at most  $(\frac{d_{avg}}{2R})N$  in both cases.*

This can be proven by allowing an adversary to choose the delay or speed ratio. The probability of conflict is maximized when the delay is zero and the speeds are the same (proof omitted).

In addition, the algorithm satisfies several other constraints that occur in large distributed systems. For example, the algorithm does not require global IDs, or global knowledge of  $N$  or the diameter of the network.

**Processor Failures:** The clubs algorithm is robust to non-leader processor failures because processors execute relatively independently and the communication is simple. Leaders guarantee local routing and diameter. The failure of a leader may potentially disconnect the group and increase the diameter. The clubs algorithm can be extended so that whenever a leader fails, its followers rerun the clubs algorithm to elect a new leader(s). Each leader periodically reasserts its leadership to indicate it is alive. The overlap in groups provides further robustness by decreasing the number of orphaned followers. Details of the timeouts parameters and correctness are provided in [10]. Thus, clubs can reorganize to accommodate failures as well as newly added processors.

### 3.1 Physical Properties of Clubs

A distinctive property of an amorphous computer is that it has topology as well as geometry derived from the communication radius and the embedding of the amorphous computer. The geometry can be used to derive additional properties of the clubs algorithm. Assuming that there are *no leadership conflicts* and processors are embedded in a two dimensional plane, we can derive two bounds on the clubs algorithms:

**Theorem 4:** *The maximum number of clubs formed is fixed for a given surface area and communication radius, and does not depend on  $N$ .*

**Theorem 5:** *In the club graph, where clubs are nodes and overlapping regions are edges, the degree of a club is at most 24. A follower can belong to no more than 9 clubs.*

Since non-adjacent leaders are at least distance  $r$  apart, both theorems can be restated as packing problems for circles of radius  $r/2$  (proof omitted). If the number of leadership conflicts is small, these theorems still hold with high probability as is shown by the simulation results presented. These bounds are useful for designing algorithms that use clubs. For example, the second bound indicates that the processors can be decomposed into groups of small diameter such that the graph induced by the groups has small degree. This is the basis for several distributed algorithms [5, 6].

### 3.2 Simulation Results

The simulation results correspond well with the analysis. We simulated an amorphous computer running the clubs algorithm with 1000, 4000 and 8000 processors, each with average neighborhood sizes of 10, 30 and 50, for different values of  $\alpha$ . The processors are uniformly distributed over a unit square surface and are asynchronous with a small delay. The processors choose a random value from the range  $[0, R)$  where  $R = \alpha d_{avg}$ . Graph 2 plots the average number of leadership con-

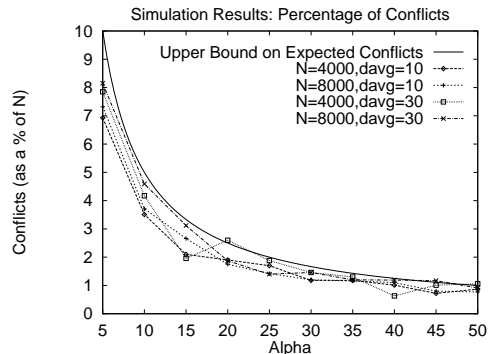


Figure 2: LEADERSHIP CONFLICTS VS. ALPHA

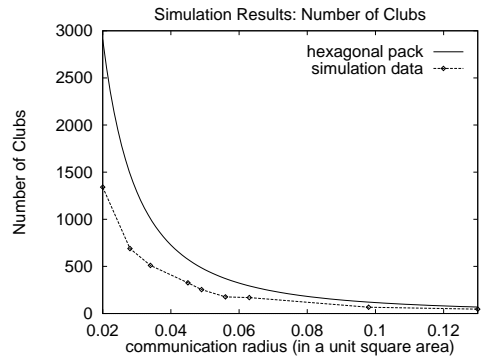


Figure 3: NUMBER OF CLUBS VS. RADIUS

licts, as a percentage of  $N$ , for four  $(N, d_{avg})$  pairs<sup>1</sup>. The average percentage of conflicts varies as expected with  $\alpha$  and does not seem to be affected by  $N$  or  $d_{avg}$ . Graph 3 plots the average number of clubs formed in the same experiments against the communication radius. Even with up to 10% conflicts, the total number of clubs varies with the radius as expected.

## 4 Extensions

In this section we describe the extension of the clubs algorithm to solve for a maximal independent set (MIS) and  $\Delta + 1$  vertex coloring. A MIS is a set of nodes in a graph such that no two nodes in the set are adjacent (independent) and no nodes can be added to that set without violating independence (maximal). Computing a MIS of a network is useful for solving many distributed computing problems [9, 8]. A  $\Delta + 1$  vertex coloring is a graph coloring that uses at most  $\Delta + 1$  colors, where  $\Delta$  is the maximum degree of the graph.  $\Delta + 1$  coloring is closely related to MIS [8].

A set of club leaders with no leadership conflicts constitutes a MIS on an amorphous computer. Leadership conflicts can be removed by running a new round of clubs on only those processors that experienced con-

<sup>1</sup>The remaining curves also occupy the same region, so for clarity we have plotted only 4 of the 9 curves

flicts in the previous round. Since in each round, at most a constant fraction,  $(1/2\alpha)$ , of the competing processors are expected to have conflicts, the number of rounds of clubs is expected to be  $O(\log N)$ . This is comparable to algorithms presented in [8]. However, the clubs algorithm uses fewer messages per round, avoids collisions and  $\alpha$  can be chosen to reduce the number of rounds and hence reduce the total cost of synchronization between rounds, which is significant in an asynchronous amorphous environment.

In order to achieve  $\Delta + 1$  coloring, the clubs algorithm is modified so that there are no leaders or followers. When a processor's countdown reaches zero, it simply chooses the smallest color it has not heard broadcasted and broadcasts that color. If there are any conflicts due to two adjacent processors choosing colors at the same time, only those processors need to run a new round of color-picking. The algorithm has similar running time to that for finding the MIS.

**Theorem 6 :** *The expected time to find the MIS or produce a  $\Delta + 1$  coloring is  $O(d_{max} \log N)$  in a synchronous amorphous computer, where  $d_{max}$  is the maximum neighborhood size. (Proof omitted)*

## 5 Example Applications

Clubs can be used for task specialization, increased robustness, or resource allocation. In this section we provide examples of using the clubs to address the issue of efficient communication in an amorphous computer. Several other examples are presented in [6, 10].

The clubs can be used as a higher level point-to-point network where the leaders communicate point-to-point with each other and relay messages to and from their members. The communication between adjacent leaders is accomplished via elected representatives in the overlap regions, significantly reducing potential collisions. For example, a full broadcast operation can be implemented by constructing a spanning tree on the graph induced by the leaders. Alternatively, a cellular network can be organized by using a coloring algorithm to assign different channels to overlapping clubs, requiring at most 24 distinct channels (Theorem 5). It is also possible to use the clubs-based  $\Delta + 1$  coloring algorithm to implement CDMA (code division multiple access) in an asynchronous amorphous computer [7]. This will create more efficient communication for applications that require frequent local exchanges of values, such as partial differential equation (PDE) calculations and cellular automata style local rules. Our hardware prototype will support spread spectrum CDMA.

## 6 Conclusion

In this paper we present the *clubs algorithm* for forming groups in an amorphous computer. The algorithm performs efficiently by relying on the local broadcast mechanism rather than point-to-point message exchanges and by keeping leadership conflicts probabilistically low. The simplicity of the local leader election mechanism allows it to perform in the asynchronous environment without loss of efficiency. The algorithm is extended to adapt to processor failures and solve traditional symmetry breaking problems such as MIS and  $\Delta + 1$  coloring. We show how the physical embedding of the amorphous computer can be used to derive additional properties of the clubs. Lastly, we present simulation results and examples of applying the clubs algorithm to address communication issues in an amorphous computer.

## References

- [1] Abelson, Knight, and Sussman. Amorphous computing. *White paper*, October 1995. <http://www-swiss.ai.mit.edu/~switz/amorphous/>.
- [2] Berlin. *Towards intelligent structures: active control of buckling*. PhD thesis, MIT, EECS Dept., May 1994.
- [3] Hall *et al.* A hierarchic control architecture for intelligent structures. *J. of Guidance, Control and Dynamics*, 14(3):503–512, 1991.
- [4] Awerbuch, Berger, Cowen, and Peleg. Fast distributed network decompositions and covers. *J. of Parallel and Distr. Comput.*, 39:105–114, 1996.
- [5] Awerbuch, Goldberg, Luby, and Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th Annual Symposium on Foundations of Computer Science*, pages 364–369, October 1989.
- [6] Coore, Nagpal, and Weiss. Paradigms for structure in an amorphous computer. AI Memo 1614, MIT, 1997.
- [7] Tanenbaum. *Computer Networks, second edition*. Prentice-Hall of India, New Delhi, 1990.
- [8] Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4), November 1986.
- [9] Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Wonderland, 1996.
- [10] Nagpal and Coore. An algorithm for group formation and maximal independent set in an amorphous computer. AI Memo 1626, MIT, 1997.