

# A PROBABILITY COLLECTIVES APPROACH TO WEIGHTED CLUSTERING ALGORITHMS FOR AD HOC NETWORKS

Geoffrey S. Ryder  
Baskin School of Engineering  
Department of Computer Engineering  
University of California - Santa Cruz  
1156 High Street  
Santa Cruz, CA, 95064-1077, U.S.A.  
gryder@soe.ucsc.edu

Kevin G. Ross  
Baskin School of Engineering  
Department of Information Systems  
and Technology Management  
University of California - Santa Cruz  
1156 High Street  
Santa Cruz, CA, 95064 - 1077, U.S.A.  
kross@soe.ucsc.edu

## ABSTRACT

The Weighted Clustering Algorithm has been proposed for mobile ad hoc wireless networks to select a subset of the nodes to be local purveyors of media access control services to the rest of the nodes. The nodes selected are called clusterheads, and each clusterhead services the other nodes within its radio range. The underlying optimal assignment problem is NP-hard. We introduce a new metaheuristic known as probability collectives to the problem of assigning clusterheads. This is an agent-based approach, with each node in the network acting in a quasi-autonomous way. This new approach shows improved clustering results, particularly with regard to the distribution and balance of cluster sizes.

## KEY WORDS

MANETs, weighted clustering algorithm, ad hoc networks, probability collectives, optimization

## 1 Introduction

In contemporary mobile wireless networking, groups of autonomous moving nodes improvise temporary local communication topologies, known as ad hoc networks. Examples include unmanned aerial vehicles, autonomous space rovers, mobile sensors, personal laptop computers, or short range communicators belonging to disaster recovery personnel or soldiers on the battlefield. It is an inefficient use of power, battery life, and bandwidth for every node to communicate with a central switch directly as a link layer entity. Instead, in the case of mobile ad hoc networks (MANETs), a few of the nodes are selected to be clusterheads, and act as purveyors of media access control or link layer services to the rest of the nodes. This establishes a two-layer hierarchy of clusterheads, and non-clusterheads (or simply cluster members). Only the clusterheads themselves must suffer the increased power consumption required for communication outside of the cluster.

In cases where all nodes are equally capable transmitters, selecting the clusterhead is critical for power manage-

ment. Choosing the best assignment possible benefits all the nodes by providing coverage for all at the lowest cost from the point of view of the entire group. Unfortunately, choosing the optimal clustering assignment is an NP-hard problem [1]. This work concerns methods for solving the assignment problem using a new, advanced optimization technique known as probability collectives.

### 1.1 The Weighted Clustering Algorithm

The Weighted Clustering Algorithm (WCA) was proposed by Chatterjee, et al. [2] for choosing clusterheads. In WCA, a node is selected to be a clusterhead when it minimizes a function of: the number of potential members of this node's cluster (limited by its effective radio range); the sum of the distances to other members; this node's mobility (where less movement is desired); and time spent thus far as a clusterhead, as a proxy for remaining battery life.

Both simulated annealing [3] and genetic algorithms [4] have been applied to this problem, and were compared to a greedy election algorithm. Both methods gave improved results, particularly in a load balancing metric, in which both types of optimized clustering assignments had much less variance in the cardinality of the clusters as compared to the greedy assignment. Given that reported improvements due to simulated annealing and genetic algorithm methods were about the same, and that in similar applications a probability collectives solver was shown to outperform genetic algorithms [5], it was decided to compare probability collectives to simulated annealing in the simulation results for this paper.

### 1.2 Probability Collectives

The method of probability collectives (PC) solves an optimization problem by turning it into a convex problem in a space of probability distributions. In this setting it can be used like simulated annealing, genetic algorithms, and other metaheuristics as an *incomplete* solver for an NP-hard combinatorial optimization problem, with the

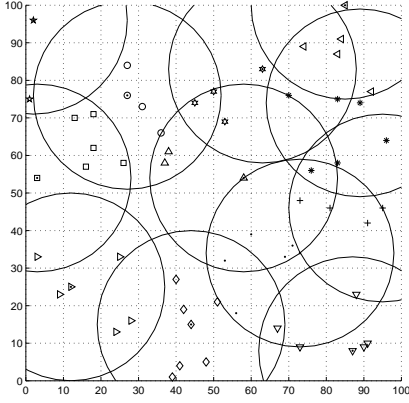


Figure 1. Weighted Clustering Algorithm (WCA) greedy election method used to make cluster assignments for a field of 60 wireless mobile network nodes. Maximum radio ranges are shown by the large circles. Members of different clusters are indicated by different symbols.

goal of getting a high-quality solution. By contrast, due to the curse of dimensionality a *complete* solver for large instances of such a problem may suffer impractically large run times.

A PC solver operates on a set of agents or decision variables. The  $\{0, 1, \}$  values indicate whether a particular node is selected as a clusterhead. In PC, instead of deciding a definite move (0 or 1) for each node, the agents allocate a *probability* to each move. At each iteration, the individual nodes act somewhat independently, updating their own probability of being selected based on a knowledge of the prior states at all other nodes.

## 2 Three Clustering Algorithms

We compare three cluster allocation algorithms based on the cost assignment metric described in [2]. Upon initialization, or occasionally as nodes travel out of range of their current cluster, the state of the field of nodes is sampled once. The assignment calculation is assumed to be much faster than changes due to the nodes' movement, so a single sample is enough. This state is the location and weight  $W_v$  for each node  $v \in N$ . Here  $N$  is the set of all nodes for which the clustering topology will be computed, and  $|N|$  is the total number of nodes. Two examples of the solutions to the assignment problem are illustrated in Figure 1 and Figure 2.

The WCA takes into account four relevant factors for each potential clusterhead node. It defines the value  $W_v$  as a cost metric to be minimized – good choices for clusterhead nodes have small weights. It follows that a good set of clusterheads which covers the entire collection of nodes will have a small value for its total weight. The term *dominant set* describes a set of special clusterhead nodes whose cluster areas together include all nodes  $N$ , i.e., a dominant set is a feasible solution to the clustering assignment prob-

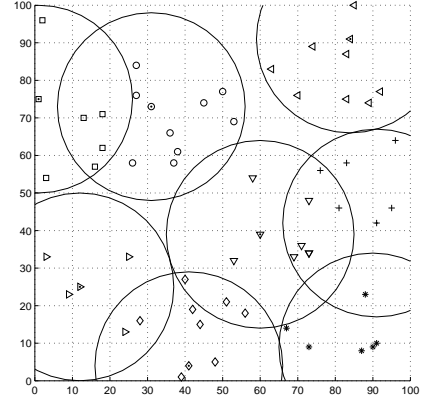


Figure 2. This is the same example as Figure 1, but the choice of clusters needed to cover all the nodes has been optimized by a probability collectives (PC) solver. Relative to the WCA greedy algorithm, simulations showed that for  $|N| = 60$  nodes, PC gave a 17% lower cost on average over all trials. On a measure of how evenly nodes are distributed, called the load balance factor (LBF), PC was 6.33 times better.

lem; and  $s$  denotes that dominant set. To see if a node  $v$  qualifies to be in  $s$ , its fitness score  $W_v$  is determined as follows:

$$W_v = w_1 \Delta_v + w_2 D_v + w_3 M b_v + w_4 P_v \quad (1)$$

The weights  $w_i$  are constrained to be positive fractions, and they must sum to one. The goal is to minimize the value of the sum of all the clusterheads' weighted costs,  $\sum_{v \in s} W_v$  over all possible dominant sets  $s$ . In Equation 1,  $\Delta_v = (M_v - |N_v|)$ , where  $M_v$  is the maximum number of other nodes that this node can support with media access control services, and  $|N_v|$  is the number of other nodes within radio range. The cost function penalizes large values for  $\Delta_v$  as they indicate underutilization of clusterhead capabilities. The term  $D_v$  is the sum of the distances between node  $v$  and potential members of its cluster,  $N_v$ .  $M b_v$  is the mobility of the node. Slowly moving nodes make better clusterheads as they cause fewer reaffiliations of cluster members, hence fewer recalculations of the clusterhead assignment. Finally  $P_v$  is the accumulated time that this node has been a clusterhead. Acting as a clusterhead is an extra power drain on a node, as compared to a more passive cluster member. Where nodes are battery-powered, setting  $w_4$  higher will help preserve the remaining battery life of the nodes.

For published experiments [2, 3, 4], the weights assigned were  $w_1 = 0.7$ ,  $w_2 = 0.2$ ,  $w_3 = 0.05$ , and  $w_4 = 0.05$ . This weighting emphasizes creating as few clusterheads as possible to cover the space, with each clusterhead running at high efficiency. The MAC support capacity  $M_v$  was set at 10 for all nodes, and experiments were run on problem instances of 20 to 60 nodes in total. To make the number of variables manageable we hold some parameters fixed. All problems have a radio range with an isotropic radius equal to one quarter of the

100x100 field where the nodes are randomly placed, with distance units arbitrary. This forces the clustering method to choose among various alternatives. Each problem instance is considered to be a snapshot of the topology at the time the clustering algorithm is invoked. The optimal reaffiliation of all the nodes is sought for that single sample.

With the nature of the problem thus limited, the number of reaffiliations per unit time, and the number of dominant set updates per radio range size are not considered here. Note that some recent results building on WCA have emphasized mobile node simulations over time, such as work on adaptive clustering based on particle swarm optimization [6], and on physical characteristics of nodes [7]. One comparable metric that this study shares with them is the number of clusterheads  $|s|$  versus  $|N|$ , the number of nodes, described in Section 3.

In general, it is preferable to have balanced cluster sizes. The **load balance factor** (LBF) in equation 2 was proposed in [3].

$$LBF = \frac{|s|}{\text{Var}_{v \in s} |N_v|} \quad (2)$$

Here  $|s|$  is the number of clusterheads, and  $|N_v|$  is the number of members of  $v$ 's cluster. If the variance of cluster membership is zero, as desired, the LBF will be infinite.

## 2.1 The Greedy Election Method

One method for assigning a dominant set  $s$  is a straightforward greedy election [2]. Considering each node  $v$  in  $N$  to be a potential clusterhead, the  $W_v$  values for all nodes are sorted in order from lowest to highest; node  $v$  with the lowest  $W_v$  is selected as the clusterhead; all its neighbors in radio range  $N_v$  are made members of its cluster; and the entire group of nodes  $\{v, N_v\}$  is removed from the sorted election list. This process is repeated until every node is assigned to be either a clusterhead or a cluster member. The election process is fast and gives fairly low-cost clustering assignments. However, the result can be suboptimal over the dominant set's total weight  $\sum_{v \in s} W_v$ ; and additionally the result may give an unbalanced grouping of nodes.

## 2.2 The Weighted Clustering Algorithm Optimized with Simulated Annealing

To find an improved, smaller total weight and larger LBF than the greedy election could provide, Turgut et al. showed how to apply the simulated annealing (SA) method to this minimization problem [3]. SA does a random walk over the solution space, here the set of potential clusterhead sets, always accepting moves that improve the objective function, and accepting moves that worsen the objective function with a finite probability  $e^{-\frac{|\Delta_{obj}|}{T}}$ . This is useful in the case of a complex objective with many local minima; in the early stages of the run,  $T$  takes a

WCA-SA				
$T_0$	$\alpha_T$	$L$	ST	$\epsilon$
50	0.9	100	10	0.1
WCA-PC				
$T_0$	$\alpha_T$	$N_{MC}$	ST	$\epsilon$
50	0.98	30	10	0.1

Table 1. Summary of the parameters used for simulated annealing and probability collectives methods.

high value and the algorithm can avoid local minima. As the temperature  $T$  is annealed (gradually decreased), the algorithm's propensity to explore is reduced. At the end of the run, SA typically *freezes* at a desirable low value of the objective, which is in practice close to (not necessarily exactly) the global minimum [8].

The outline of the WCA-SA algorithm is shown below. Let  $s$  be one choice of dominant set (a set of nodes that are clusterheads), and let the fitness of that dominant set be  $F(s) = \sum_{v \in s} W_v$ . Then  $s^*$  and  $F^*$  are the optimal values found so far. See Table 1 for the constant values, which in the case of SA are the same as those used in prior work.  $L$  is the number of sample solutions checked before reducing the temperature  $T$ . Constant  $\alpha$  is the fractional multiplier used to reduce  $T$  at each iteration. Constant  $\epsilon$  determines convergence: if the difference between  $F^*$  values at successive temperatures is less than  $\epsilon$  for  $ST$  consecutive annealing steps, WCA-SA terminates.

### WCA-SA

```

1 Initialization: select a random dominant set s
2 Compute F(s)
3  $s^* \leftarrow s; F^* \leftarrow F(s); T \leftarrow T_0$ 
4 While (frozen==FALSE)
5   Repeat L times
6     Select random neighboring solution  $s'$ 
7     If ( $F(s') < F(s)$ )
8        $s^* \leftarrow s'; F^* \leftarrow F(s')$ 
9     Else
10       $X \leftarrow$  uniform random number in [0,1]
11      If ( $X < e^{-\frac{|\Delta_{obj}|}{T}}$ )
12         $s^* \leftarrow s'; F^* \leftarrow F(s')$ 
13       $T \leftarrow \alpha_T * T$ 
14      if ( $\Delta F(s) < \epsilon$  for  $ST * L$  consecutive steps)
15        frozen  $\leftarrow$  TRUE
16 End

```

In the subroutine for line 6, successive trial solutions should be neighbors. Since it is not clear from previous results what neighborhood structure was utilized, we created the two-part random neighbor generating scheme below. We select the set  $N_d$  (with  $|N_d| \in [1, 4]$ ) to be a random set of clusterheads that will be dropped in this iteration. When a clusterhead is dropped, all the member nodes it used to support will lose their affiliation and seek a new assignment. So, given a dominant set  $s$ ,

### A. New Dominant Set Random Neighbor

```

1 Randomly delete the subset  $N_d \subset s$  clusterhead nodes from  $s$ 
2 Add all  $N_d$  deleted clusterhead nodes to a tabu list  $t_d$ 

```

3 Randomly choose nodes from newly unassigned nodes,  $v_n \notin t_d$ , to be new clusterheads until all nodes are again covered

### B. New Node Cluster Assignment

1 **For** every node  $v_n$  still unassigned  
 2     **Repeat** for all clusterheads  $v$  in range of node  $v_n$   
 3         Find closest  $v$  by Euclidean distance  
 4         **If**  $v$  has remaining capacity,  $\Delta_v > 0$ ,  
 5             Assign  $v_n$  to its cluster  
 6             break;  
 7         **Else**  
 8             for  $v_n$ , disqualify  $v$ 's cluster  
 9 Remaining unassigned  $v_n$  become clusterheads

This pair of sequential subroutines is used for line 6 in the WCA-SA algorithm. Part B is needed because although at that point a valid dominant set  $s'$  exists, there can be overlap for a dense field of nodes, and so for each unassigned member node a choice of alternative clusterheads must be made. The new dominant set  $s'$  combines considerable structure from the original  $s$ , and also a random component that allows SA to explore the space of potential solutions. One advantage of Part B over the original WCA method is that the boundaries of each cluster do not limit the selection. Previously, the finest-grained assignment was done at the level of the cluster. Now it is done at the level of the node – the element determining the affiliation is the individual unassigned node, which adds flexibility in choosing solutions.

## 2.3 The Weighted Clustering Algorithm Optimized with Probability Collectives

Here we provide an introduction to probability collectives (PC) theory, with more details available in [9, 10, 11, 12]. PC Theory's roots come from statistics, machine learning and game theory, where ideas about how to harness the collective intelligence of many-agent systems to solve problems have been developed. In such a system, agents select actions, and receive rewards based on the system objective function. What sets PC Theory apart from other optimization methods is that it operates directly on probability distributions instead of individual solutions. Let  $x_v$  be the event that node  $v$  is a clusterhead, set to one if it is selected as part of the set  $s$  or zero otherwise. In PC theory, instead of assigning 1 and 0 to  $x_v$  a probability  $q_v(x)$  defines the *probability* that  $x_v$  is set to the value  $x$ , where  $x \in \{0, 1\}$ . Each iteration of PC updates the  $q_v(x)$  values, with the objective of moving toward a set of probabilities that assign values close to 1 for the optimal  $x_v$  values. Since this application deals with a binary decision, there are only two probabilities for each node, and the decision constitutes a Bernoulli random variable with probability  $q_v(1) = 1 - q_v(0)$ . For simplicity in notation we use  $q_v$  to denote  $q_v(1)$ .

One key to PC theory is the method of updating probabilities. This is done by an independent decision at each

node. Each node is aware of the previous  $q_u$  values for the other nodes  $u \neq v$ , and updates its own  $q_v$ . From the perspective of each node, the immediate objective is to minimize

$$g_v(q_v) = E_q[G(x_v, x_{(v)})|q_v] \quad (3)$$

$$= q_v E_q[G|x_v = 1] + (1 - q_v) E_q[G|x_v = 0]$$

where  $(v)$  denotes the set of all nodes apart from  $v$ ,  $G(x)$  is the global objective function (here the total dominant set weight), and the expectations  $E_q[G|x_0]$  are taken over the other agents' previous distributions. Equation (3) represents node  $v$ 's contribution to the global objective. At any point in time, the agents are acting independently, hence the joint distribution is a product distribution, leading to the relationship

$$E_q[G|x_v] = \sum_{x_u \in X_{(v)}} \prod_{u \neq v} q_u(x_u) G(x_u, x_v) \quad (4)$$

where  $X_{(v)}$  is the set of all possible values for  $x_u, u \neq v$ . To efficiently find the global minimum without focusing on extreme point solutions, PC theory creates a convex optimization problem over the space of the nodes' distributions. This is done by introducing an entropy term according to the maximum entropy principle (maxent): when given incomplete prior information about a distribution  $q_v$ , the estimate of  $q_v$  should contain the minimal amount of extra information beyond that already known a priori about  $q_v$ . The entropy term of each Bernoulli distribution is a scalar quantity, found by computing  $p * \log(\frac{1}{p})$ , and summing:

$$S(q_v) = q_v \ln(\frac{1}{q_v}) + (1 - q_v) \ln(\frac{1}{1 - q_v}) \quad (5)$$

This term measures the spread of the distribution of outcomes for node  $v$ , with a minimum at 0.5 (the widest distribution) and approaching zero as the probability moves toward selecting a node with probability one or zero.

To find the minimum value of  $G$ , we form the maxent Lagrangian equation for each agent, written

$$\mathcal{L}_v(q_v) = g_v(q_v) - TS(q_v) \quad (6)$$

One purpose of introducing entropy is to create a convex probability space over the agent's distributions that has a single minimum. The maxent Lagrangian is convex over the set of product distributions over the agents' move space [9]. By operating on  $q_v$  in this convex space we are able to use powerful search methods for finding function extrema developed for continuous domain problems, such as gradient descent. Here we are iteratively adjusting  $q_v$  to agree with the iteratively computed gradient that points to the bottom of the convex maxent Lagrangian "bowl". Note that while adding entropy makes the descent easier, it also biases the method away from extreme (hence true) solutions. That bias is gradually lowered by annealing  $T$ .

In [11], several search methods for descending the maxent Lagrangian are derived. In this paper we use the nearest Newton update formula shown below in Equation 8, which was demonstrated to provide high quality results and to converge quickly.

$$K_{update} = \frac{E[G|x_v] - E[G]}{T} + S(q_v) + \ln(q_v(x_v)) \quad (7)$$

$$q_v(x_v) = q_v(x_v) - \alpha_{step} q_v(x_v) * K_{update} \quad (8)$$

Recall that  $G$  is computed as  $G = F(s) = \sum_{v \in s} W_v$ . Since it is impractical to calculate all outcomes of the product distribution,  $E[G]$  is estimated by considering a set of  $N_{MC}$  Monte Carlo samples of the underlying agent distributions. Further,  $E[G|x_v]$  can also be estimated over the same set of MC samples with the value of  $x_v$  substituted in as node  $v$ 's move for every sample vector.

### 2.3.1 PC Implementation Details

The following outline shows the structure of the PC implementation, which is similar to that given in [10].

#### WCA-PC

```

1 Set  $q_v$  to be uniform over  $\{0,1\}$ 
2  $T \leftarrow T_0$ 
3 While (frozen==FALSE)
4   For each  $v \in N$ 
5     Generate  $N_{MC}$  random samples of  $s$  from  $q_v$  values
6     Evaluate  $E[G]$  using the random samples
7      $s^* \leftarrow$  best solution from the random samples of  $s$ 
8   For each  $v \in N$ 
9     Evaluate  $E[G|x_v = 1]$  and  $E[G|x_v = 0]$ 
       using random samples
10    Update  $q_v$  using Nearest Newton rule
11     $T \leftarrow \alpha_T * T$ 
12    if ( $\Delta F(s) < \epsilon$  for ST consecutive steps)
13      frozen  $\leftarrow$  TRUE
14 End

```

The most computations are done in step 9. There, each agent estimates of  $E[G|x_v = 0]$  and  $E[G|x_v = 1]$  by taking the Monte Carlo outcomes for all other agents. This incurs a large run time since for each move of each agent, the  $F(s)$  value of a solution is computed  $N_{MC}$  times. As a result of step 9, the term  $E[G|x_v] - E[G]$  is available for update step 10. Both steps 6 and 9 use the new node cluster assignment method, Part B, shown before for WCA-SA, to evaluate the cost of each Monte Carlo solution vector. This gives a fair comparison between the methods.

## 3 Simulation Study

The three algorithms described in Section 2 were implemented and tested, and the results are summarized in Table 2. The settings and results given in prior work [2, 3, 4, 7] were our benchmarks, in which it is desirable for the cost to be minimized and the LBF to be maximized. Each entry in the table is an average of 20 runs; for each value of  $N$ , all four methods were tested using the same 20 data sets to

Metric	$ N $	WCA	Descent	SA	PC
Cost,	20	82.70	76.30	71.80	70.00
Cost,	40	132.08	117.19	116.73	110.61
Cost,	60	179.80	164.76	165.40	148.51
LBF,	20	0.05	0.50	0.66	0.70
LBF,	40	0.04	0.21	0.18	0.28
LBF,	60	0.03	0.09	0.08	0.19
Clusters,	20	8.75	7.05	7.10	7.35
Clusters,	40	11.15	7.00	8.40	8.85
Clusters,	60	12.30	9.20	9.50	10.4

Table 2. Mean values of simulation results, over 20 data sets. Cost is  $F(s)$ , LBF is the load balance factor, and Clusters is the number of clusters,  $|s|$ .

obtain valid comparisons. SA is WCA-SA, PC is WCA-PC, and *Descent* is the same as WCA-SA, but without the annealing feature of accepting weaker solutions. Simple descent did quite well, implying that the solutions were not plagued by poor local minima. The performance of WCA-PC was best, and improved relative to the other methods as the problem size grew. For  $|N| = 20$ , valid choices of nodes for the dominant set were limited, and the differences among methods were small. At  $|N| = 60$ , there were many choices and the PC solver's results were notably improved. The number of clusters in Table 2 is comparable to previous results [7].

From one annealing step to another, the benchmark WCA-SA calls the new node assignment function 100 times (Part B. of the random selection subroutine) while WCA-PC calls it  $2 \cdot N_{MC} \cdot N + N_{MC} = 2430$  times in the 40 node case. Yet WCA-PC's results are available after the first 30 calls, as shown in Figure 4. The remaining computations are for training the agent distributions. The run time is determined by the time it takes for each method's annealing while-loop to stop. By this criteria, WCA-PC can take longer than SA. But as shown in Figure 4, WCA-PC can deliver a better solution quickly without waiting for annealing steps. The key factor for WCA-PC is the size of the Monte Carlo block,  $N_{MC}$ , which here is set to 30. This is fairly low among published benchmark values for  $N_{MC}$ .

## 4 Conclusion and Future Work

We have introduced a new method, probability collectives, to solve the problem of grouping MANET nodes into optimal clusters using the WCA criteria. It has shown very promising performance compared with benchmark algorithms such as simulated annealing. Ongoing research will evaluate the impact of mobility, tighter power constraints and fully distributed control. With its adaptive learning technique, PC theory is naturally suited to these extensions.

Compared to the other methods, on the average WCA-PC gave the best results for cost and for LBF for the benchmark study. Each of these methods has many parameters that can be adjusted, and we expect different methods to perform favorably under different conditions. The probability collectives approach deserves serious consider-

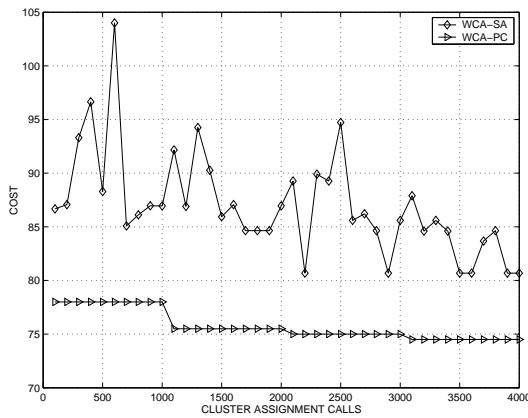


Figure 3. This plot shows the performance over time of a single problem instance solved by SA and PC on a 20 node benchmark case. These are intermediate values just shown to illustrate how the solution progresses. For WCA-PC, this is the mean value from the 30 most recent Monte Carlo samples. For WCA-SA, this is the best value found from the previous 100 solutions.

ation as a metaheuristic for difficult applications such as the Weighted Clustering Algorithm.

## Acknowledgment

This research is supported by NASA through the University Affiliated Research Center. The authors thank David Wolpert, William Macready, and Stefan Bieniafski for their introduction to PC Theory, and the anonymous reviewers for their insightful suggestions.

## References

- [1] S. Basagni, I. Chlamtac, & A. Farago, A generalized clustering algorithm for peer-to-peer networks, *ICALP, Proc. of the Workshop on Algorithmic Aspects of Communication*, Bologna, Italy, 1997.
- [2] M. Chatterjee, S.K. Sas, & D. Turgut, An on-demand weighted clustering algorithm (WCA) for ad hoc networks, *Proc. 43rd IEEE Global Telecommunications Conference*, San Francisco, CA, 2000, 1697-1701.
- [3] D. Turgut, B. Turgut, R. Elmasri, & T.V. Le, Optimizing clustering algorithm in mobile ad hoc networks using simulated annealing, *Proc. IEEE Wireless Communication and Networking Conference*, New Orleans, LA, 2003, 1492-1497.
- [4] D. Turgut, S.K. Das, R. Elmasari, B. Turgut, Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithm approach, *Proc. 45th IEEE Global Telecommunications Conference*, Taipei, Taiwan, 2002, 62-66.
- [5] C.F. Huang, D.H. Wolpert, S. Bieniafski, & C.E.M. Strauss, A Comparative Study of Probability Collectives Based Multi-Agent Systems and Genetic Algorithms, *Proc. of the Genetic and Evolutionary Computation Conference*, Washington, DC, 2005, 751-752.

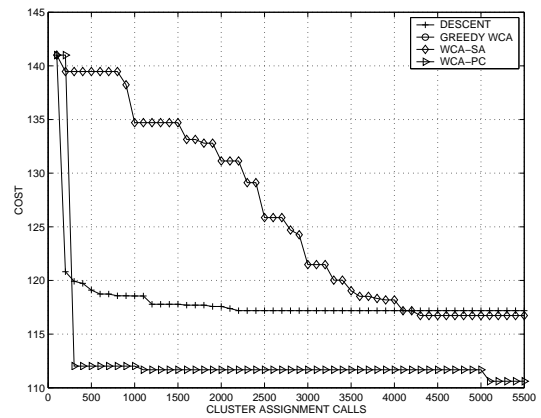


Figure 4. This plot shows the average performance of each algorithm over the same problem sets for the 40 node benchmark case. The x axis has been scaled to the number of new node cluster assignment calls. Both WCA-SA and WCA-PC use the same subroutine to determine the solution cost  $F(s)$ . After one block of Monte Carlo samples, or 30 assignment calls, WCA-PC results are available. WCA-SA results become available after every block of 100 assignment calls.

- [6] C. Ji, Y. Zhang, S. Gao, P. Yuan, & Z. Li, Particle swarm optimization for mobile ad hoc networks clustering, *Proc. IEEE International Conf. on Networking, Sensing and Control*, Taipei, Taiwan, 2004, 372-375.
- [7] S.K. Dhurandher & G.V. Singh, Weight based adaptive clustering in wireless ad hoc networks, *IEEE International Conference on Personal Wireless Communications*, New Delhi, India, 2005, 95-100.
- [8] K. Manousakis, A.J. McAuley, & R. Morera, Applying simulated annealing for domain generation in ad hoc networks, *IEEE International Conference on Communications*, Paris, France, 2004, 3864-3868.
- [9] D.H. Wolpert & S.R. Bieniafski, Distributed control by Lagrangian steepest descent, *Proc. 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004, 1562-1567.
- [10] S.R. Bieniafski, I.M. Kroo, & D.H. Wolpert, Discrete, continuous, and constrained optimization using collectives, *Proc. 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, 2004, 4850(1-9).
- [11] D.H. Wolpert, Finding bounded rational equilibria part II: alternative Lagrangians and uncountable move spaces, *11th International Symposium on Dynamic Games and Applications*, Tucson, AZ, 2004, T7.
- [12] D.H. Wolpert & C.F. Lee, Product distribution theory for control of multi-agent systems, *Third Intl. Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, 2004, 522-529.