Improving Network Connectivity in Ad Hoc Networks Using Particle Swarm Optimization and Agents

Abdullah Konak, Orhan Dengiz, and Alice E. Smith

Abstract In a mobile ad hoc network (MANET) the nodes can serve as both routers and hosts and can forward packets on behalf of other nodes in the network. This functionality allows the MANET to form an instant, autonomous telecommunication network without an existing infrastructure or a central network control. This chapter introduces a dynamic MANET management system to improve network connectivity by using controlled network nodes called agents. Agents have predefined wireless communication capabilities similar to the other nodes in the MANET. However, the agents' movements, and thus their locations, are dynamically determined to optimize network connectivity. A particle swarm optimization (PSO) algorithm is used to choose optimal locations of the agents during each time step of network operation.

1 Introduction

Mobile wireless ad hoc networks (MANET) are instantaneous, autonomous telecommunication networks that provide service to users wherever and whenever the service is needed. The communication depends on wireless links that are formed between the mobile nodes. A link is established between two nodes if they are within each other's wireless communication range. A MANET topology is dynamic and expected to change often and unpredictably due to arbitrary movements of its nodes. Therefore, a MANET might be disconnected intermittently during its mission time. This chapter introduces an approach to maintain global connectivity in MANETs by

Abdullah Konak
Penn State Berks, Reading, PA, USA, e-mail: auk3@psu.edu

Orhan Dengiz
DnD Technical Solutions , Ankara, Turkey, e-mail: orhan.dengiz@gmail.com

Alice E. Smith
Auburn University , Auburn, AL, USA e-mail: smithae@auburn.edu

using controlled MANET nodes, called agents. The function of an agent node is to augment network connectivity. During a mission time of a MANET, the user nodes are assumed to move freely. However, their current and past location data are available, and a mobility prediction model can be used to predict their future locations. Based on the predicted locations of user nodes, the agent nodes are directed to move to new locations to improve connectivity among the user nodes. Although networks operate in continuous time, to make the optimization tractable, the continuous time operation is sliced into T discrete time steps.

Let $G_t(N_t, E_t)$ denote a MANET with node set N_t and arc set E_t at time t. Node set N_t includes two types of nodes: user nodes (UN_t) and agent nodes (AN_t). User nodes are nodes that demand network services. Agent nodes are responsible for helping the user nodes experience the best network service possible. At the beginning of time t, it is assumed that the current and past locations (x_{ik}, y_{ik}) are known for all nodes $i \in UN_t$ and $k = 1, ..., t$. The decision problem at time t is to determine the best new location $(x_{i(t+1)}, y_{i(t+1)})$ for each agent node $i \in AN_t$ to maximize a measure of network connectivity. Note that it will take some time for agent nodes to move from their current locations at the beginning of time t to their new locations, and during this time user nodes may also move to new positions. Between time t and $t+1$, each agent node i moves to its new location (i.e., $(x_{i(t+1)}, y_{i(t+1)})$). It is assumed that at the beginning of each time period, new location data becomes available, and the new best locations of agent nodes should be determined again. Therefore, the decision problem should be solved as quickly as possible when new location data becomes available at the beginning of each time period, and the decisions about agents' next locations should be propagated to them.

This chapter presents a particle swarm optimization (PSO) algorithm to direct the motion of the mobile agents in MANETs where the global state of the network can be tracked using a GPS tracking system. In GPS tracking systems, each node is equipped with a GPS receiver, and the location data is periodically transmitted to a central location using low frequency radio or a satellite modem embedded in the node [18]. The proposed approach in this chapter includes a mobility prediction model to predict the future locations of the nodes, and the PSO algorithm determines the new locations of the agent nodes based on this prediction. The proposed PSO algorithm is tested under dynamic scenarios using a simulation environment. Among application areas of the proposed approach are network-centric warfare, fleet tracking, and search and rescue operations.

2 Background

In the literature, several approaches have been proposed to address the challenges in MANETs due to unpredictable user node movements. One of the major problems is the accessibility of the centralized network services used by all network nodes when the network is disconnected. A solution approach to this problem involves replicating network services [32, 33] and critical data [19] at multiple nodes and

dynamically deploying these nodes to disconnected partitions of the network. Another problem is delivery of data packets across disconnected network partitions. To address this problem, a few papers [10, 34] propose using special agent nodes that can buffer packets until their destination nodes are reachable. These agent nodes may move arbitrarily or systematically, and when they are connected to a network partition, they deliver their payloads. This approach, however, is mainly applicable to delay-tolerant data networks. Alternatively, several papers propose topology control by modifying node trajectories or power level nodes. In [23], the intermediate nodes between a source and a destination node modify their trajectories to ensure the delivery of packets, and an algorithm to minimize the trajectory modifications is proposed. Goyal and Caffery [15] propose first determining critical links whose failures cause partitioning of the network, and then supporting these links by either modifying the trajectory of the nodes involved in the critical links or bringing an outside node to reinforce them. Kim et al. [22] use transmission power management schemes to strengthen the network topology across the critical nodes of a network.

There has been very limited work in the literature to improve network connectivity in MANETs through mobile agents. Ou [25] uses forwarding nodes that can adjust their locations to assist network partitions in an ad hoc network. When deployed initially, forwarding nodes move randomly to discover network partitions. It is assumed that each node has GPS capability and the nodes can exchange location data. When a forwarding node receives a service request, it will move to the service area if it is available (i.e., it is not servicing other nodes). However, the paths of the forwarding nodes are not optimized in this approach. Chadrashekar et al. [6] define the problem of achieving connectivity of a disconnected ground MANET by dynamically placing unmanned air vehicles (UAVs) which function as relay nodes. Then, they develop a heuristic algorithm for the single-UAV problem. Zhu et al. [28] also propose using UAVs equipped with communication capabilities to provide services to ground-based MANETs. They define a gradient-based algorithm to determine the location of a single UAV to maximize two network connectivity measures, global message connectivity and worst-case connectivity. Both connectivity measures are defined on a spanning tree so the optimization problem is to determine the most reliable Steiner tree of the network, where the UAV is considered a Steiner point in the network. Recently, Hauert et al. [16] propose the deployment of a swarm of UAVs for search and rescue missions. During a mission, UAVs are expected to maintain direct or indirect connection to their base-station through the ad hoc network that they form. An ant-colony approach [11] is developed to control the UAV MANET.

3 Proposed MANET Management System and Problem Description

The user nodes in the MANET move at their own will and it is assumed that their future positions are unknown. However, the location data of users and agents are assumed to be available to the agent control system at all times. Tracking user and

agent node locations is technically possible by the GPS tracking technology discussed in Section 1. Arc set E_t at time t depends on the locations of the nodes and their transmission ranges as follows: $E_t = \{(i,j) : \min(R_{it}, R_{jt}) \geq d_{ijt} \quad \forall i,j \in N_t\}$ where R_{it} is the wireless transmission range of node i and d_{ijt} is the Euclidean distance between nodes i and j at time t. Each arc (i,j) has a transmission capacity u_{ijt} (in bits per second-bps), which is also a function of the distance between nodes i and j at time t.

3.1 Objectives and Evaluating Network Connectivity

The overall objective is to increase the connectivity of a MANET. In the ideal case, there should be at least one path between every user node pair in the network. Therefore, the primary objective is to connect each user node to every other one. This overall connectivity objective can be expressed as:

$$O_{1t} = 2 \times \frac{\sum_{i,j \in UN_t : i<j} z_{ijt}}{nu_t \times (nu_t - 1)} \tag{1}$$

where $z_{ijt} = 1$ if there is a path between user nodes i and j at time t, and $z_{ijt} = 0$, otherwise, and nu_t is the number of user nodes at time t. Intuitively, equation (1) is the percentage of user node pairs that are connected at time t.

Equation (1) is a meaningful objective function if the network is disconnected. For a connected network, however, equation (1) is not sensitive with respect to changes in agent node locations. In other words, equation (1) will have the value of one for a possibly infinite number of agent node location combinations as long as the network is connected. However, each of these combinations may result in different network topologies, link capacities and network performance. A good measure to quantify the connectivity of a connected network is the capacity of the cut set with the minimum capacity among all cut sets of the network. Let $MaxFlow(G_t, i, j)$ be the maximum data (in bps) that can be sent from node i to node j through all possible paths between these two nodes on network G_t. $MaxFlow(G_t, i, j)$ provides a good measure of the connectivity between nodes i and j, as well as the quality of the links between them. $MaxFlow(G_t, i, j)$ can be calculated by solving a maximum flow problem with node i as the source and node j as the sink on capacitated network G_t. The minimum capacity cut set of a network can be determined by solving $MaxFlow(G_t, i, j)$ for all node pairs i and j of the network. However, solving the maximum flow problem is a computationally expensive operation, particularly for large problems. Since the network operates dynamically in real time, a good solution must be quickly found during each time step so that agent nodes can be properly deployed to their new locations. To reduce the computational time, the maximum flows between clusters of nodes, instead of between individual nodes, are considered in the approach herein. Let $C_{1t}, ..., C_{Kt}$ denote K clusters of user nodes at time t. The second criteria to gauge network connectivity is given as:

$$O_{2t} = \min_{\substack{i=1,\ldots,K-1,\\ j=i+1,\ldots,K}} \left\{ MaxFlow(G_t, C_{it}, C_{jt}) \right\} \tag{2}$$

In the calculation of $MaxFlow(G_t, C_{it}, C_{jt})$, first a dummy node d_i is connected to all nodes of cluster C_{it} using directional arcs with unbounded capacities and a dummy node d_j is connected to all nodes of cluster C_{jt} in a similar fashion, and then the maximum flow between d_i and d_j is calculated. O_{2t} improves the weakest connection between the clusters of user nodes. In addition, it is a responsive objective function for connected networks. For disconnected networks, O_{2t} is not required to be evaluated since it is always zero. In summary, O_{1t} is used to compare two solutions with disconnected topologies, and O_{2t} is used to compare two solutions with connected topologies.

3.2 Future User Location Prediction

The objectives defined in the previous section are considered for each time step t. However, the proposed system is expected to maximize network connectivity during a mission time of T time periods. Therefore, predicting the future locations of user nodes and deploying agent nodes based on these predictions may improve the average connectivity during the mission time. In the literature, there are a few papers addressing the location prediction problem in MANETs. Wang and Chang [31], Su et al. [29] and Tang et al. [30] utilize a simple location and velocity based expected position to help the routing protocols in MANETs. Ashbrook and Starner [2] propose a Markov model to predict the next important station which a user will be located at. This approach is only valid over the specific area that is used for learning. Mitrovic [24] develops a neural network model to predict car motion. This neural network is trained using specific maneuvers on certain road conditions. However, the proposed neural network approach can be adapted for more generalized motion patterns. Creixell and Sezaki [9] report promising results to predict pedestrian trajectories using a time series method. Kim et al. [21] propose a double-exponential smoothing approach to predict velocity and direction of mobile users. In a more recent study, Huang and Zaruba [17] propose a method that enables non-GPS equipped ad hoc nodes to estimate their approximate locations by using information from GPS equipped nodes.

Any of the location prediction methods cited above can be used in the proposed MANET management system, but in this chapter the double-exponential smoothing approach is selected since it is computationally efficient and it can be used to predict node velocities. The predicted location of user node i at time $t+H$ is given as:

$$\hat{x}_{i(t+H)} = \bar{x}_{it} + Hv_{it}^{x}$$
$$\hat{y}_{i(t+H)} = \bar{y}_{it} + Hv_{it}^{y}$$

where $\bar{x}_{it}$ and $\bar{y}_{it}$ are the smoothed locations of user node i along the x-axis and y-axis, respectively, and v_{it}^x and v_{it}^y are the corresponding estimated velocities. $\bar{x}_{it}$ and v_{it}^x are updated based on the past locations as follows:

$$\begin{aligned} \bar{x}_{it} &= \alpha_1 x_{it} + (1-\alpha_1)\bar{x}_{i(t-1)} \\ v_{it}^x &= \alpha_2(\bar{x}_{it} - \bar{x}_{i(t-1)}) + (1-\alpha_2)v_{i(t-1)}^x \end{aligned}$$

where α_1 and α_2 are smoothing parameters between 0 and 1 for the expected location and velocity, respectively. Note that $\bar{y}_{it}$ and v_{it}^y are also updated in a similar fashion.

3.3 Optimization Problem and Deployment Decision

At the beginning of any time t, the locations of user nodes (UN_t) and agent nodes (AN_t) are known. The overall decision problem is to relocate the agents from their current positions at time t to their new locations at time $t+1$. However, considering predicted user locations further in the future than the next time period and deploying the agents accordingly may improve the connectivity of the network over the long term. Therefore, the optimization problem is formulated to determine the future locations of the agents to maximize network connectivity at time $t+H$. In other words, the PSO algorithm searches for the best location of each agent node i at time $t+H$. Let $(x_{i(t+1)}, y_{i(t+1)})$ denote the deployment decision of agent node i at time t. The distance that an agent node can travel during a time period is limited by $D_{\max}$, and the deployment decision of an agent node at a time period determines its starting location at the following time period. Therefore, the optimization problem at time t is expressed as follows:

Problem $ALOC(t,H)$:

$$Max \quad z = O_{1(t+H)} + O_{2(t+H)}$$

$$\sqrt{\left(x_{i(t+H)} - x_{it}\right)^2 + \left(y_{i(t+H)} - y_{it}\right)^2} \le H \times D_{\max} \quad \forall i \in AN_t$$

The outcome of problem $ALOC(t,H)$ is the best location $(x^*_{i(t+H)}, y^*_{i(t+H)})$ of each agent node i at time $t+H$. However, due to the dynamic nature of the problem, at time $t+1$, which is the start of a new optimization cycle, the new location information of user nodes becomes available and the optimization problem will be solved again with this new information. Therefore, the outcome of problem $ALOC(t,H)$ must be implemented during the single time period between t and $t+1$. Based on $(x^*_{i(t+H)}, y^*_{i(t+H)})$, the deployment decision at time t is as follows:

$$\begin{aligned} x_{i(t+1)} &= x_{it} + (x^*_{i(t+H)} - x_{it})/H \\ y_{i(t+1)} &= y_{it} + (y^*_{i(t+H)} - y_{it})/H \end{aligned} \tag{3}$$

Equation (3) means that each agent node i is deployed at time t with a commitment to be at location $(x^*_{i(t+H)}, y^*_{i(t+H)})$ at time $(t+H)$ if the direction and velocity of the agent are not changed. Note that the velocities and directions of agent nodes are assumed to be constant during their deployment between times t and $t+1$.

4 Particle Swarm Optimization and Implementation

4.1 Particle Swarm Optimization (PSO)

PSO is a population-based meta-heuristic which emulates the social behavior of species that live in the form of swarms in nature. These swarms are capable of exchanging valuable information such as food locations in the habitat. PSO was developed by Eberhart and Kennedy [12]. In PSO, the aim of particles is to search for the optimal point in a continuous search space as they constantly move from one point to another according to their velocities. For a problem with n decision variables, each particle j in swarm S is represented by three vectors $\mathbf{X}_j = \{x_{1j}, ..., x_{nj}\}$, $\mathbf{P}_j = \{p_{1j}, ..., p_{nj}\}$, and $\mathbf{V}_j = \{v_{1j}, ..., v_{nj}\}$. Vector $\mathbf{X}_j$ represents the current position of particle j, $\mathbf{P}_j$ represents the position of the particle's historic best fitness, and $\mathbf{V}_j$ is the velocity vector that defines the direction and magnitude of the particle's movement in the search space. $\mathbf{V}_j$ is used to update $\mathbf{X}_j$ each iteration as follows:

$$\mathbf{X}_j \leftarrow \mathbf{X}_j + \mathbf{V}_j \tag{4}$$

As particles swarm over the search space, they communicate with each other, broadcasting their best found positions and learning about the best found positions of others. Based on these interactions as well as their own past experiences, particles adjust their velocities in each iteration as follows:

$$\mathbf{V}_j \leftarrow \omega \mathbf{V}_j + \phi_1 \cdot (\mathbf{P}_j - \mathbf{X}_j) + \phi_2 \cdot (\mathbf{G}_{N(j)} - \mathbf{X}_j) \tag{5}$$

where $N(j)$ is the set of particles which particle j shares information with (i.e., the neighborhood of particle j) and $\mathbf{G}_{N(j)} = \{g_{1j}, ..., g_{nj}\}$ is the best position discovered so far by the particles in $N(j)$. ϕ_1 and ϕ_2 are vectors of n uniform random numbers between 0 and φ_1 and between 0 and φ_2, respectively. Parameters φ_1 and φ_2 are called cognition and social coefficients, respectively, and are very important for facilitating convergence in PSO. According to an empirical study [26] as well as theoretical results [8], it is recommended to set φ_1 and φ_2 such that $\varphi_1 + \varphi_2 \leq 4$ in order to keep the growth in velocity magnitudes under control. Note that the operator $(\cdot)$ in equation (5) denotes element-by-element vector multiplication.

Parameter ω is called the inertia constant and is used to balance local versus global search. In a standard PSO, ω is usually set to one, which encourages exploration of new areas in the search space. For small values of ω such as $\omega < 0.4$, the search shifts to the intensification mode as particles tend to oscillate between $\mathbf{P}_j$ and

$\mathbf{G}_{N(j)}$. Dynamically updating ω, starting with a value close to one for global search at the initial stages of the search and then gradually reducing it for local search at the final stages, has been shown to improve the performance of PSO algorithms [27]. Selecting a proper neighborhood structure $N(j)$ depends on the problem type. A common approach is to use a global neighborhood structure where each particle shares information with every other particle in the swarm (i.e., $N(j) = S$). The global neighborhood structure leads to fast convergence, but it may increase the chance of being trapped at a local optimum.

The pseudo code of the standard PSO is presented in Fig. 1 for a maximization objective function $f()$. In the pseudo code, $U(a,b)$ represents a random sample from the uniform distribution on $[a,b]$ and $\mathbf{G}^* = \{g_1, ..., g_n\}$ is the best position found so far by all particles. The initial positions of particles are randomly determined between the maximum and minimum values of each decision variable ($x_i^{\max}$ and $x_i^{\min}$, respectively). Alternatively, particles can also be systematically initialized in the search space. Particle velocities are also randomly initialized between the maximum and minimum velocity limits ($v_i^{\max}$ and $v_i^{\min}$, respectively). Velocity limits usually depend on the ranges of the decision variables.

```
for each particle X_j ∈ S do {
   x_ij ← U(x_i^min, x_i^max)   for i = 1,...,n
   v_ij ← U(v_i^min, v_i^max)   for i = 1,...,n
   P_j ← X_j
   if f(X_j) > f(G_N(j)) then G_N(j) ← X_j
   if f(X_j) > f(G*) then G* ← X_j
}
while(Stop is false){
   for each particle X_j ∈ S do {
      for i = 1,...,n do {
         v_ij ← v_ij + U(0,φ1)(p_ij − x_ij) + U(0,φ2)(g_ij − x_ij)
         x_ij ← x_ij + v_ij    }
      if f(X_j) > f(G_N(j)) then G_N(j) ← X_j
      if f(X_j) > f(P_j) then P_j ← X_j
      if f(X_j) > f(G*) then G* ← X_j
   }}
return G*
```

Fig. 1 Pseudo code of the standard PSO.

PSO has successfully been applied to problems in the continuous domain. It has few parameters that require adjustment, which makes the development process relatively easy and fast. Implementation is also easy due to its simple but robust mechanism. PSO has also been applied to dynamic problems. Eberhart and Shi [13] test PSO on nonlinear optimization problems where the optimal points change during the search. They show the ability of PSO to track changing objectives. Carlisle and Dozier [5] modify the memory property (**P** vector maintenance strategy) of

the swarm for dynamically changing environments. When a change in the problem environment is detected, all memory positions (**P**) are reevaluated and set to either the old memory or to the current particle position, whichever is better. Their results show that PSO can successfully track a time dependent objective function.

4.2 The Optimization Procedure

In order to solve Problem ALOC(t,H) using PSO, each particle j in swarm S is represented by three vectors: $\mathbf{X}_{jt} = \{(x_{ijt}, y_{ijt}) : i \in AN_t\}$, $\Delta\mathbf{X}_{jt} = \{(\Delta x_{ijt}, \Delta y_{ijt}) : i \in AN_t\}$, and $\mathbf{X}^*_{jt} = \{(x^*_{ijt}, y^*_{ijt}) : i \in AN_t\}$. Vector $\mathbf{X}_{jt}$ represents the current position of particle j in the solution space at time t, vector $\mathbf{X}^*_{jt}$ is the position of particle j with the best fitness found, and $\Delta\mathbf{X}_{jt}$ is the velocity vector that defines the direction and magnitude that the particle will travel in the solution space if not disturbed. The particle velocities are updated in each iteration as follows:

$$\begin{aligned} \Delta x_{ijt} &\leftarrow \omega\Delta x_{ijt} + U(0,\varphi_1)(x^*_{ijt} - x_{ijt}) + U(0,\varphi_2)(x^*_{it} - x_{ijt}) \quad \forall i \in AN_t \\ \Delta y_{ijt} &\leftarrow \omega\Delta y_{ijt} + U(0,\varphi_1)(y^*_{ijt} - y_{ijt}) + U(0,\varphi_2)(y^*_{it} - y_{ijt}) \quad \forall i \in AN_t \end{aligned} \tag{6}$$

where (x^*_{it}, y^*_{it}) denotes the location of agent node i according to the best solution found so far ($\mathbf{X}^*_t$) during the search at time t. In other words, the PSO uses a global neighborhood to update the particle velocities. The PSO employs a dynamic inertia coefficient ω. In this chapter, ω is initially set to 1.5 and decreased geometrically by a coefficient of 0.98 at each iteration. Although the common practice is to let the inertia coefficient decrease monotonically, from the preliminary experimentation it was found beneficial to occasionally reset it to its initial value. This improves the ability to escape local optima by "exciting" the particles every now and then, helping them swarm to other regions. The inertia coefficient is reset to its original value of 1.5 with a probability of 0.02, for each particle at each iteration step. With a relatively high inertia coefficient, the current direction and magnitude of the particle's motion are weighted heavily. As the iterations advance and the inertia coefficient is decreased, changing the direction and magnitude of the particle velocities toward $\mathbf{X}^*_{jt}$ and $\mathbf{X}^*_t$ becomes easier. The swarm particle velocities are limited within $[-D_{\max}, D_{\max}]$. After updating particle velocity $\Delta\mathbf{X}_{jt}$, the new position of each particle $j \in S$ is calculated as follows:

$$\begin{aligned} x_{ijt} &\leftarrow x_{ijt} + \Delta x_{ijt} \quad \forall i \in AN_t \\ y_{ijt} &\leftarrow y_{ijt} + \Delta y_{ijt} \quad \forall i \in AN_t \end{aligned} \tag{7}$$

Once new position vector $\mathbf{X}_{jt}$ is determined for particle j, the objective functions are evaluated for the new position, and $\mathbf{X}^*_{jt}$ and $\mathbf{X}^*_t$ are updated accordingly. The first step in the evaluation of $\mathbf{X}_{jt}$ is to form capacitated network $G_t(N_t, E_t)$. Then, O_{1t} can be calculated. If $O_{1t}(\mathbf{X}_{tj}) = 1$, then all user nodes are connected and O_{2t} is evaluated. To do so, K clusters of user nodes are determined using a clustering algorithm based on the Kruskal Spanning Tree Algorithm (see pages 515-516 of [1]). Then, the

maximum flows between each node cluster pair are computed and the minimum of them is used as O_{2t}. There can be cases where two different disconnected networks have the same value of O_{1t} (i.e., $O_{1t}(\mathbf{X}_{jt}) < 1$ and $O_{1t}(\mathbf{X}_{jt}) = O_{1t}(\mathbf{X}^*_{jt})$). In such cases, a tertiary criterion is used to break ties between two disconnected networks with the same value of O_{1t} as follows. First a depth-first search is used to identify all disconnected partitions of the network such that user nodes in a network partition are connected, but they are not connected to the nodes in other network partitions. Then, the midpoint between the closest nodes of two network partitions is calculated for each partition pair. These midpoints are called attraction points since deploying the agent nodes close to the attraction point between two disconnected partitions may connect them. Finally, the distances between agents and attraction points are calculated and the minimum of these distances is used as the tie breaker between two disconnected networks with the same O_{1t}. This tertiary criterion is expressed as follows:

$$O_{3t} = \min_{\substack{i \in AN_t \\ j \in A_t}} \{\sqrt{(x_{it} - x_{jt})^2 + (y_{it} - y_{jt})^2}\} \tag{8}$$

where A_t is the set of attraction points at time t. The particle evaluation and update procedure of the PSO is given in Fig. 2.

```
Procedure EvaluateUpdateParticle()
Evaluate O_1t(X_jt)
if O_1t(X_jt) < 1 and O_1t(X_jt) > O_1t(X*_jt) then X*_jt ← X_jt
if O_1t(X_jt) < 1 and O_1t(X_jt) = O_1t(X*_jt) then{
      Evaluate O_3t(X_jt)
      if O_3t(X_jt) < O_3t(X*_jt) then X*_jt ← X_jt
      }
if O_1t(X_jt) = 1 then{
      Evaluate O_2t(X_jt)
      if O_2t(X_jt) > O_2t(X*_jt) then X*_jt ← X_jt
      }
if O_1t(X*_jt) + O_2t(X*_jt) > O_1t(X*_t) + O_2t(X*_t) then X*_t ← X*_jt
```

Fig. 2 Solution evaluation procedure of the PSO algorithm.

At the beginning of a time step t, the inputs to the PSO algorithm are: (i) the best deployment decision from the previous interval (i.e., $\mathbf{X}^*_{t-1} = \{(x^*_{i(t-1)}, y^*_{i(t-1)}) : \forall i \in AN_{t-1}\}$), and (ii) the predicted user node locations at time $t+H$. The deployment decisions of the agents at time $(t-1)$ become their starting locations at time t. Each particle j is initialized as described in procedure InitializeSwarm() given in Fig. 3.

Procedure InitializeSwarm()
for each particle $\mathbf{X}_j \in S$ **do** {
$x_{ijt} \leftarrow x^*_{i(t-1)} + U(-H \times D_{\max}, HD_{\max}) \quad \forall i \in AN_t$
$y_{ijt} \leftarrow y^*_{i(t-1)} + U(-H \times D_{\max}, HD_{\max}) \quad \forall i \in AN_t$
$\Delta x_{ijt} \leftarrow U(-D_{\max}, D_{\max}) \quad \forall i \in AN_t$
$\Delta y_{ijt} \leftarrow U(-D_{\max}, D_{\max}) \quad \forall i \in AN_t$
}

Fig. 3 Particle initialization procedure of the PSO algorithm.

4.3 Semi-Intelligent Agent Node Behavior

The agent location optimizer requires agent nodes to be an integral part of the MANET in order to improve the objective functions. Since the main objective is to improve the experience of user nodes, agent nodes are not included in O_{1t} and O_{2t}. This means that an isolated agent node, which is an agent node with a node degree of zero or one, will have no effect on the objective functions. Note that to maximize O_{1t} and O_{2t} agent nodes must have two or more arcs. In dynamic settings, however, agent nodes can become isolated due to sudden changes in the directions and velocities of user nodes. The PSO algorithm may not guide an isolated agent to better locations since the objective functions are insensitive to small changes in the positions of isolated agents. If an agent becomes disconnected from the network, it has to be able to move independently and rejoin the MANET again to be properly utilized. A semi-intelligent behavior of agent nodes is adopted as described below.

Assume that for position $\mathbf{X}_{jt}$, particle j has an agent node i which is isolated at time t. Velocity Δx_{ijt} of this agent node is reset as follows:

$$\begin{aligned} \Delta x_{ijt} &\leftarrow x_{A_{ijt}} - x_{ijt} \\ \Delta y_{ijt} &\leftarrow y_{A_{ijt}} - y_{ijt} \end{aligned} \tag{9}$$

$x_{A_{ijt}}$ and $y_{A_{ijt}}$ are the x and y coordinates of the attraction point with respect to the agent node i in $\mathbf{X}_{jt}$. The attraction point for an isolated agent i depends on whether the network is connected or not. If the network is connected, then the attraction point is the last geometric center of the network and if the network is disconnected then it is the closest A_t that is used in the calculation of O_{t3}. By resetting the velocity of isolated agents, they are stimulated to rejoin the rest of the network. Considering this semi-intelligent behavior the overall pseudo procedure of the PSO algorithm is given in Fig. 4. In the pseudo code, $c_{\max}$ represents the maximum number of the PSO iterations during a time period. Inertia constant ω is randomly reset to its original value by a particle with a probability of 0.02.

```
Procedure PSO()
InitializeSwarm()
c ← 1; ω ← 1.5
while (c < c_max){
   c ← c + 1
   ω ← 0.98ω
   for each particle j ∈ S do {
      Update ΔX_jt using (6)
      Calculate new particle position X_jt using (7)
      EvaluateUpdateParticle()
      Reset the velocity of isolated agents using (9)
      if U(0,1) < 0.02 then ω ← 1.5
   }
}
Return X*_t
```

Fig. 4 Overall procedure of the PSO algorithm.

5 Computational Experiments

The proposed PSO algorithm is tested using both dynamic and static scenarios. For the dynamic scenarios, the movements of user nodes are simulated using a random mobility model over a mission of T time periods. The details of this mobility model are given in section 5.1. To evaluate the performance of the MANET management system and the PSO algorithm in dynamic scenarios, two performance metrics are used based on the objective functions of the problem as follows:

$$\begin{aligned} P_1 &= \frac{\sum_{t=1}^{T} (O_{1t})}{T} \\ P_2 &= \frac{\sum_{t=1}^{T} \min_{i,j \in UN_t : j>i} \{MaxFlow(G_t,i,j) : MaxFlow(G_t,i,j) > 0\}}{T} \end{aligned} \tag{10}$$

Performance metrics P_1 and P_2 are indicators of the algorithm performance over the entire simulation time span T. P_1 is the average user connectivity ratio and P_2 is the average all-pair minimum bandwidth in bits per second during the simulation time span. In dynamic scenarios, the performance of the PSO algorithm and the proposed MANET management system are tested as described in the procedure given in Fig. 5. Note that Problem $ALOC(t,H)$ is solved for the predicted locations of user nodes, and performance metrics P_1 and P_2 are evaluated using the actual locations of user nodes. Similarly, the clusters of user nodes are also determined in each time period based on the predicted locations of user nodes. A minimum spanning tree-based clustering algorithm (see [1] pages 515-516 for details) is used to identify K clusters of user nodes. First, the minimum spanning tree of user nodes is determined in the full mesh network of user nodes, where the rectilinear distances between user nodes are used as the arc weights, and then K disconnected clusters of user nodes are obtained by deleting (K-1) arcs of the minimum spanning tree one

by one in decreasing order of their distances. The maximum flows between the pairs of user nodes (or user clusters) are calculated by an implementation of the highest-label push-relabel algorithm [14] available from BOOST C++ libraries, which is a peer-reviewed, freely available software library collection [3]. The PSO code was implemented in the C programming language and all runs were performed on Linux PCs with Dual 3.0 GHz Intel Xeon E5450 Quad-Core Processors and 32 GB RAM.

for $t = 1, \ldots, 3$ **do** $\{$
  Using the mobiliy model, generate locations $(x_{it}, y_{it}) \quad \forall i \in UN_t$
$\}$
for $t = 3, \ldots, T$ **do** $\{$
  Predict $(\hat{x}_{i(t+H)}, \hat{y}_{i(t+H)}) \quad \forall i \in UN_t$
  Determine K clusters, $C_{1t}, ..., C_{Kt}$, based on the predicted user locations
  Solve Problem $ALOC(t,H)$ to determine $(x^*_{i(t+H)}, y^*_{i(t+H)}) \quad \forall i \in AN_t$
  Deploy mobile agent i to $(x_{i(t+1)}, y_{i(t+1)}) \quad \forall i \in AN_t$
  Simulate actual locations $(x_{i(t+1)}, y_{i(t+1)}) \quad \forall i \in UN_t$
  Create network $G_{(t+1)}$
  Evaluate $G_{(t+1)}$
$\}$
Calculate P_1 and P_2

Fig. 5 Procedure for testing dynamic problems.

5.1 Mobility Simulation Environment

To test the performance of the PSO algorithm under dynamic scenarios, a mobility simulation model was created to randomly generate the movements of user nodes. In the simulation model, each user node i is assigned to a random starting point (x_{i0}, y_{i0}) and to a random destination point (x_{iT}, y_{iT}), and each follows a path with random perturbations to its destination. Let $\mathbf{v}_{it}$ be the unit vector representing the direction of the motion of user node i at time t. At each time period, each user node i travels a random distance $U(d_{\min}, d_{\max})$ in the direction of vector $\mathbf{v}_{it}$ which is calculated as follows:

$$\mathbf{v}_{it} \leftarrow \begin{cases} \left(\alpha_v \mathbf{v}_{i(t-1)} + (1-\alpha_v)\frac{(x_{iT},y_{iT})-(x_{it},y_{it})}{|(x_{iT},y_{iT})-(x_{it},y_{it})|}\right)\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} & \text{if } U(0,1) < p, \\ \alpha_v \mathbf{v}_{i(t-1)} + (1-\alpha_v)\frac{(x_{iT},y_{iT})-(x_{it},y_{it})}{|(x_{iT},y_{iT})-(x_{it},y_{it})|} & \text{otherwise.} \end{cases}$$

where α_v is a weight factor for the current motion direction, θ represents the random changes in the direction of the motion, and p determines how frequently the direction is perturbed. The initial directions of user nodes are also randomly deter-

mined. This motion simulation model is a version of the Random Waypoint Model [4], which is frequently used in the literature as a benchmark mobility model to evaluate MANET management protocols. By changing parameters $d_{\min}$, $d_{\max}$, α_v, θ, and p, very complex user motions can be achieved. The simulation parameters for the test problems are: $d_{\min} = 0.01$, $d_{\max} = 0.05$, $\alpha_v = 0.95$, $\theta = U(-\pi/4, \pi/4)$, and $p = 0.1$ This motion scenario models the case where users search for their destinations or make their way around forbidden areas or obstacles, which is a reasonable representation of a search/rescue or a military operation.

Although the proposed MANET management system and the PSO algorithm can handle cases where the number of user and agent nodes change during the simulation, it is assumed that the number of agent and user nodes is constant during the simulation. The simulation area is a two-dimensional rectangular area with lower-left corner coordinates of $(0,0)$ and upper-right corner coordinates of $(5,5)$. The simulation dimensions are set so that the wireless transmission ranges of all MANET nodes are one unit distance ($R_{it} = 1.0$). Without loss of generality, normalized transmission ranges and data rates are used in the simulation. Using the Euclidian distance d_{ijt} between two nodes i and j at time t, the link capacity u_{ijt} in terms of data rate in bits per second (bps) is approximated with a continuous function as follows:

$$u_{ijt} = 10^4 \left(1 + e^{10 \cdot (d_{ijt} - 0.5)}\right)^{-1} \tag{11}$$

Static test problem data specifies the locations of user nodes, which are randomly selected within the simulation area.

5.2 The Effect of Number of Agents, Future Location Prediction, and Clustering

To study the effect of the input parameters of the proposed MANET system, two test problem groups were used. Five 10-user and five 20-user dynamic test problems were randomly generated with $T = 100$ as described in the test problem simulation environment. Each random problem was solved five times using the PSO algorithm with different random number seeds for the MANET system input parameter values of the number of agents $na = \{1,2,3\}$, the number of prediction horizons $H = \{0,1,2,3,4,5,6\}$, and the number of clusters $K = \{4,6\}$. For problems with 10 user nodes, node clustering was not used (i.e., $K = 10$). In other words, 25 replications were performed for each combination of na, H, and K (a total of 1050 runs for the 20-user group and 525 for the 10-user group). The swarm size was 40 in all runs and the PSO algorithm was run for $c_{\max} = 50$ iterations during each time step of the simulation. In real-world cases, parameter $c_{\max}$ depends on how frequently the problem is solved and deployment decisions are propagated to agent nodes. An ANOVA model was used for each problem group to study the effect of na, H, and K on the performance of the proposed MANET management system.

Figures 6 and 7 present the main effect plots for the 10-user and 20-user problem groups, respectively. As expected, both network connectivity (P_1) and minimum bandwidth (P_2) significantly improve with the increasing number of agents. As seen in the main effect plots, location prediction has a significant effect on the performance of the network (ANOVA p-values of significance P_1: 0.0 for both problem groups; P_2: 0.065 for 10-node and 0.020 for 20-node problems). Based on these results, predicting future locations of user nodes is particularly important for improving overall network connectivity. Network connectivity performance metric P_1 significantly improves from $H = 0$ (no prediction) to $H = 1$ (prediction for the next time period). For the test problems studied, H=3 or 4 provides the best results. It should be noted that although several hundred runs were performed for the ANOVA, the results of this experiment should not be generalized. H is a parameter that should be selected by network managers based on empirical data and experience. Various mobility models or prediction techniques may yield different results. Nonetheless, the results of this experiment demonstrate that incorporating a user location prediction model into the proposed MANET management system may improve overall network performance in dynamic problem scenarios.

Another interesting result of the experiment with dynamic problems is that the number of user node clusters K has no statistically significant effect on either P_1 or P_2 (p-values of 0.3 and 0.14, respectively). The main justification for using clusters of user nodes instead of individual nodes in evaluating the weakest connection of a network is computational tractability, particularly for dynamically solving large problems. Using a cluster of user nodes makes the evaluation of O_{t2} independent of the number of user nodes. Therefore, very large networks might be efficiently evaluated in limited time periods.

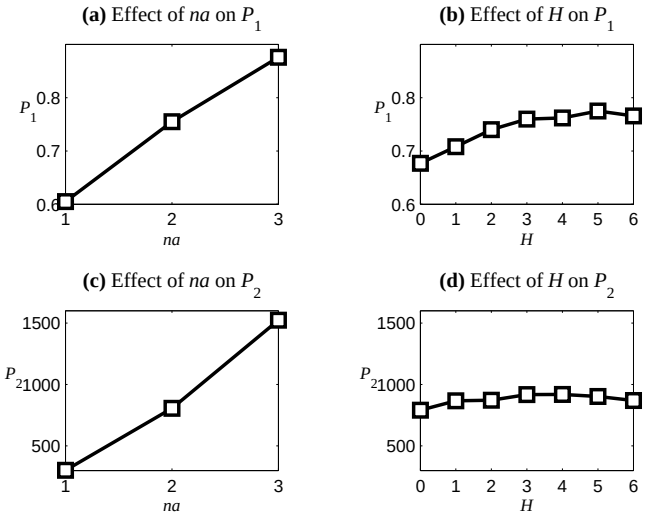

Fig. 6 Main effect plots for 10-user problems.

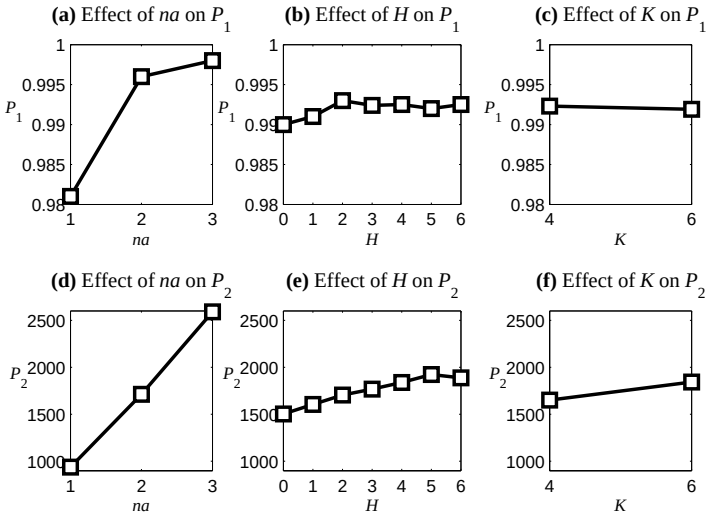

Fig. 7 Main effect plots for 20-user problems.

5.3 Comparative Performance of the PSO Algorithm

In this study, the performance of the PSO algorithm is compared with respect to a MIP formulation in several static test problems. The agent deployment problem, Problem $ALOC(t,H)$, is a very difficult non-linear MIP problem to solve to optimality. An approximate MIP formulation of the static problem was developed by only considering O_{2t} as given below (Problem *ALOC*). To do so, the nonlinear arc distance versus capacity relationship and the Euclidean arc distances were modeled using eight piecewise linear approximations. Although the MIP formulation is computationally intractable for real-world sized problems, the main objective of this comparison is to provide a benchmark for the performance of the PSO algorithm.

Because Problem *ALOC* is for static cases, time index t is omitted in the decision variables and problem parameters. In addition, the upper bound ($D_{\max}$) constraint on the distance that agent nodes can travel is irrelevant for static cases. Problem *ALOC* is a multi-commodity network flow problem where variable f_{ijst} represents the flow from the source node s to the sink node t on arc (i,j). Node set N includes both user nodes (UN) and agent nodes (AN). Arc set E includes two types of arcs, user node arcs (i.e., $\{(i,j) : \min(R_i,R_j) \geq d_{ij} \quad \forall i,j \in UN, i \neq j\}$) and agent node arcs (i.e., $\{(i,j) : \forall i \in N, \forall j \in AN, i \neq j\}$). Because the locations of user nodes are known (let $(\hat{x}_i,\hat{y}_i)$ represent the location of user node i for all $i \in UN$), the capacities of the user node arcs are constant in Problem *ALOC* due to constraints (24) and (25). On the other hand, the capacities of agent node arcs depend on the locations of agent nodes.

The objective function of Problem *ALOC* is to maximize the minimum of the maximum flows between the user node pairs. Given the locations of the user nodes $(\hat{x}_i, \hat{y}_i)\ \forall i \in UN$, the decision variables of Problem *ALOC* include the locations of agent nodes (i.e, $(x_i, y_i)\ \forall i \in AN$). Constraints (13)-(15) are the node-flow balance constraints to calculate the maximum flow U_{st} from the source node s and to the sink node t for $\forall s,t \in UN$ and $s < t$. Constraints (16)-(20) are used to calculate the squared-Euclidian distances sd_{ij} of agent arcs (i, j) for $i \in N, j \in AN, i < j$. In constraint (20), squared-distances $(d_{ij}^x)^2$ and $(d_{ij}^y)^2$ along the x and y axes, respectively, are approximated by piecewise linear functions while solving the problem in CPLEX. Constraints (21) and (22) determine the capacity u_{ij} of each agent arc (i, j) based on (11), which is also approximated by piece-wise linear functions. Constraint (23) ensures that the flow variables do not exceed the arc capacities.

Problem *ALOC*:

$$\begin{aligned}
\text{Mazimize } z &= U \\
st. & \\
U &\le U_{st} \quad \forall s,t \in UN, s < t && (12)\\
\sum_{\{j:(i,j)\in E\}} f_{ijst} - \sum_{\{j:(j,i)\in E\}} f_{jist} &= 0 \quad \forall i \in N, \forall s,t \in UN, i \neq s, i \neq t, s < t && (13)\\
\sum_{\{j:(s,j)\in E\}} f_{sjst} - \sum_{j:(j,s)\in E\}} f_{jsst} &= U_{st} \quad \forall s,t \in UN, s < t && (14)\\
\sum_{\{j:(t,j)\in E\}} f_{tjst} - \sum_{\{j:(j,t)\in E\}} f_{jtst} &= -U_{st} \quad \forall s,t \in UN, s < t && (15)\\
d_{ij}^x &\ge x_i - x_j \quad \forall i \in N, j \in AN, i < j && (16)\\
d_{ij}^x &\ge x_j - x_i \quad \forall i \in N, j \in AN, i < j && (17)\\
d_{ij}^y &\ge y_i - y_j \quad \forall i \in N, j \in AN, i < j && (18)\\
d_{ij}^y &\ge y_j - y_i \quad \forall i \in N, j \in AN, i < j && (19)\\
(d_{ij}^x)^2 + (d_{ij}^y)^2 &\le sd_{ij} \quad \forall i \in N, j \in AN, i < j && (20)\\
u_{ij} - 10^4 \left(1 + e^{10\left(\sqrt{sd_{ij}} - 0.5\right)}\right)^{-1} &\le 0 \quad \forall i \in N, j \in AN, i < j && (21)\\
u_{ji} &\le u_{ij} \quad \forall i \in N, j \in AN, i < j && (22)\\
f_{ijst} &\le u_{ij} \quad \forall (i,j) \in E && (23)\\
x_i &= \hat{x}_i \quad \forall i \in UN && (24)\\
y_i &= \hat{y}_i \quad \forall i \in UN && (25)\\
u_{ij}, x_i, y_j, f_{ijst} &\ge 0
\end{aligned}$$

Problem *ALOC* was solved by CPLEX v11 with a limit of eight hours CPU time. CPLEX was not able to find feasible solutions for problems with more than ten user nodes and five agents within the allowed CPU time limit. Fig. 8 shows the comparisons on static cases of ten randomly generated 5-user and 10-node problems with three and five agents. In these problems, the locations of users were randomly gen-

erated in a way that connecting all users were possible. In all problems, both user and agent nodes had a communication range of one unit distance. For each problem instance, the PSO was run for ten random replications with a swarm size of 40 for $c_{\max}$=1000 iterations. Therefore, the box-plots in Fig. 8 include 100 data points for the PSO and ten data points for CPLEX. The majority of 5- and 10- node problems with three agents were solved to optimality within the CPU time limit. These static problems with three agents demonstrate that the PSO performed as well as the MIP formulation. In some replications of these problems, the PSO finds slightly better solutions than the MIP formulation. These differences can be attributed to the linear approximation in the MIP formulation. As seen in Fig. 8, the PSO algorithm perform significantly better than the MIP formulation in the problems with five agents. Increasing the number of agents from three to five significantly increases the search space of the problem and these problems could not be solved to optimality within the CPU time limit. The solutions found by the PSO are superior to the best solutions found by CPLEX over eight hours of CPU time. The CPU time requirement of the PSO depended on the problem. In random 5-node problem with three agents, for example, 1000 PSO iterations require CPU times ranging from 1.3 to 80.0 seconds with an average of 31.9 seconds. For random 10-node problems with five agents, the minimum, maximum, and average CPU times are 21.9, 855.2, and 316.5 seconds, respectively. In these problems, a significant percent of the computational effort is due to calculation of maximum flows. A more efficient approach to calculating all-pairs maximum flow (e.g., the all-pairs minimum value cut algorithm [1]) may provide significant computational savings.

6 Conclusions

In this chapter, a new model is proposed to conceptualize an autonomous topology optimization for mobile ad hoc networks. The proposed approach relocates mobile agents to help maintain a suitable level of communication service in the network. The representation of wireless ad hoc network communications as network flows and optimization using a maximum flow model is a novel approach. This representation is very responsive to small changes in topology when evaluating network connectivity and performance. Also, it can be used with any signal attenuation model when calculating the data flow rates.

The dynamic nature of the problem is a challenge, but it also enables the optimizer to gain additional information by leveraging the information obtained during the optimization at previous time steps. The usefulness of predicting user locations to improve network connectivity is demonstrated. A particle swarm algorithm is developed to solve the modeled dynamic optimization problem. Computational experiments show that the particle swarm algorithm is very promising and well suited to this problem.

The proposed approach, while developed for dynamic topology optimization, easily adapts to a static scenario by increasing the agent velocity constraints. The

static scenario is useful when users want to improve an existing system of sensors or communication hubs already positioned in the field, or when designing a new static system. The proposed approach could also be used for "what-if" purposes before launching an actual network in the field. The simulation is useful to plan for the most efficient number of mobile agents to deploy under a certain scenario, and to consider cost / benefit tradeoffs.

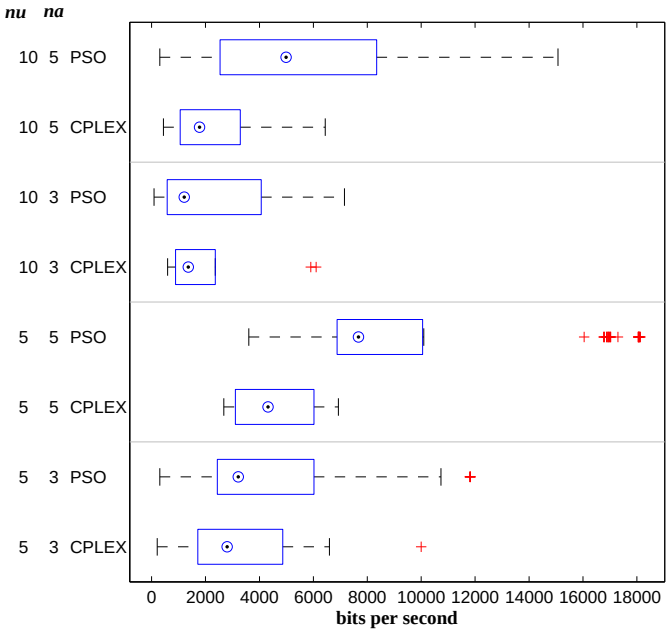

Fig. 8 Comparison of the PSO and MIP on ten random problems of each size. (*nu* = the number of users, *na* = the number of agents.)

References

1. Ahuja, RK, Magnanti, TL, and Orlin, JB: (1993) Network Flows, Theory, Algorithms, and Applications. Prentice-Hall, Inc., Upper Saddle River, New Jersey
2. Ashbrook, D, Starner, T (2002) Learning significant locations and predicting user movement with GPS. Proceedings of the 6th International Symposium on Wearable Computers (ISWC 2002): 101-108
3. BOOST.org, Boost C++ libraries, 2006, Accessed in 2006; Available at: http://www.boost.org.

4. Broch, J, Maltz, DA, Johnson, DB, Hu, YC, and Jetcheva, J (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking 85-97
5. Carlisle, A, Dozier, G (2002) Tracking changing extrema with adaptive particle swarm optimizer. Proceedings of the 5th Biannual World Automation Congress 13: 265-270
6. Chadrashekar, K, Dekhordi, MR, Baras, JS (2004) Providing Full Connectivity in Large Ad-Hoc Networks by Dynamic Placement of Aerial Platforms. The Center for Satellite and Hybrid Communication Networks, Technical Report, University of Maryland
7. Choudhury, RR, Paul, K, Bandyopadhyay, S (2002) An agent-based connection management protocol for ad hoc wireless networks. Journal of Network and Systems Management 10: 483-504
8. Clerc, M, Kennedy, J (2002) The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6: 58-73
9. Creixell, W, Sezaki, K (2004) Mobility prediction algorithm for mobile ad hoc network using pedestrian trajectory data. Proceedings of the IEEE Region 10 Conference-(TENCON 2004) 2:668-671
10. Davis, JA, Fagg, AH, Levine, BN (2001) Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. Proceedings Fifth International Symposium on Wearable Computers 141-148
11. Dorigo, M, Maniezzo, V, Colorni, A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 26: 29-41
12. Eberhart, RC, Kennedy, J (1995) A new optimizer using particle swarm theory. Sixth International Symposium on Micro Machine and Human Science :3943
13. Eberhart, RC, Shi, Y (2001) Tracking and optimizing dynamic systems with particle swarms. Proceedings of the 2001 Congress on Evolutionary Computation 1: 94-100
14. Goldberg, AV, Tarjan, RE (1988) A new approach to the maximum-flow problem. Journal of the ACM (JACM) 35:921-940
15. Goyal, D, Caffery, J (2002) Partitioning avoidance in mobile ad hoc networks using network survivability concepts. Seventh IEEE Symposium on Computers and Communications (ISCC'02) 553-558
16. Hauert, S, Winkler, L, Zufferey, J-C, Floreano D. (2008) Ant-based swarming with positionless micro air vehicles for communication relay. Swarm Intelligence 2: 167-188
17. Huang, R, Zaruba, GV (2007) Location tracking in mobile ad hoc networks using particle filters. Journal of Discrete Algorithms 5: 455-470
18. Kaplan, E, Hegarty, C (2006) Understanding GPS: Principles and Applications Second Edition. Artech House, Norwood, MA, USA
19. Karumanchi, G, Muralidharan, S, Prakash, R (1999) Information dissemination in partitionable mobile ad hoc networks. Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems 4-13
20. Kennedy, J, Mendes, R (2006) Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 36: 515-519
21. Kim, IK, Jang, SH, Lee, JS (2007) QLP-LBS: quantization and location prediction-based LBS for reduction of location update costs. Lecture Notes in Computer Science, Frontiers of High Performance Computing and Networking ISPA 2007 Workshops 4743/2007:69-76
22. Kim, T-H, Tipper, D, Krishnamurthy, P, et al. (2009) Improving the Topological Resilience of Mobile Ad Hoc Networks. 7th International Workshop on the Design of Reliable Communication Networks 191-197
23. Li, Q, Rus, D (2003) Communication in disconnected ad hoc networks using message relay. Journal of Parallel and Distributed Computing 63: 75-86
24. Mitrovic, D (1999) Short term prediction of vehicle movements by neural networks. Proceedings of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems 187-190

25. Ou, CH, Ssu, KF, Jiau, HC (2004) Connecting network partitions with location-assisted forwarding nodes in mobile ad hoc environments. The 10th Pacific Rim International Symposium on Dependable Computing 239-247
26. Ozcan, E, Mohan, CK (1999) Particle swarm optimization: surfing the waves. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 Vol. 3: 1939-1944
27. Shi, Y, Eberhart, RC (2001) Fuzzy adaptive particle swarm optimization. Proceedings of the 2001 Congress on Evolutionary Computation 1: 101-106
28. Zhu, H, Swindlehurst, AL, Liu, KJR (2006) Smart deployment/movement of unmanned air vehicle to improve connectivity in MANET. 2006 IEEE Wireless Communications and Networking Conference 252-257
29. Su, W, Lee, S-J, Gerla, M (2001) Mobility prediction and routing in ad hoc wireless networks. International Journal of Network Management 11: 3-30
30. Tang, J, Xue, G, Zhang, W (2004) Reliable routing in mobile ad hoc networks based on mobility prediction. Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems 466-474
31. Wang, N-C, Chang, S-W (2005) A reliable on-demand routing protocol for mobile ad hoc networks with mobility prediction. Computer Communications 29: 123-135
32. Wang, KH, Li, B (2002) Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. Proceedings of IEEE INFOCOM 2: 1089-1098
33. Wang, KH, Li, B (2002) Group mobility and partition prediction in wireless ad-hoc networks. Proceedings of 2002 IEEE International Conference on Communications 2: 1017-1021
34. Zhao, W, Ammar, M, Zegura, E (2004) A message ferrying approach for data delivery in sparse mobile ad hoc networks. Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing 187-198