Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp92-97)

Enhancing Concurrent Node Movements in Mobile Ad-hoc Networks while Preserving Connection Stability

JORGE BARREIROS[1,2], FERNANDA COUTINHO[1]
[1]Instituto Superior de Engenharia de Coimbra do Instituto Politécnico de Coimbra
[2]Centro de Informática e Sistemas da Universidade de Coimbra
Rua Pedro Nunes, 3030-199 Coimbra
PORTUGAL

Abstract: - In mobile ad-hoc networks loss of communication paths can occur because the movement of nodes may create network partitions, effectively preventing communication between nodes in disjoint sections. Some methods were previously proposed to specifically address this issue. For instance, connection stability may be preserved by constraining and adapting the movements of each individual node in order to ensure that there is always some viable communication path between any two nodes on the network. This is accomplished by establishing a set of waypoints each node should transverse before arriving to its desired position. The order by which the nodes move to these waypoints is specified by the algorithm. However, multiple alternate solutions for the problem can be found by varying the sub-set of nodes that moves in each step, and the degree of concurrency in the movements of the nodes is directly dependant on the choices made. In this work, we point out that there is an opportunity for applying an optimization method that will enable to maximize the number of nodes that move towards their destinations in each step, while ensuring connectivity is maintained. We present such optimization and show how it impacts overall system performance.

KeyWords: - movement coordination, mobile ad hoc networks, genetic algorithms, platooning.

1 Introduction

Mobile ad hoc networks (MANETs) are self-organizing, infrastructureless networks. MANETs are particularly promising as the communication infrastructure of mobile robotics deployed to perform some collaborative task. This is the case of a wide range of applications from the usual rescue teams for disaster area to the popular robotic football competitions.

In a MANET, nodes are able to communicate directly if they are in the transmission range of each other. If not, they must communicate using a multi-hop route. A MANET node is required to operate also as a router so that it can be able to forward data packets coming from neighbor nodes.

The movement of the networked nodes and the influence of the environment where they move make MANETs highly dynamic structures. A stable connectivity can only be achieved during small time intervals and thus hopping must be performed in an adaptive fashion. So, intensive effort has been devoted by the research community to develop efficient routing schemes for MANETs. Recent evaluations of most popular routing strategies can be found in [4, 8, 5, 9, 3]. More extensive materials discussing the subject can be obtained in [12, 11, 7].

Previous research in [1, 2] explores the possibility to coordinate the mobility of the nodes in a fashion that keeps stable connectivity among them. This method is particularly suitable in situations where, at the moment of network deployment, specific locations that should be transversed by some nodes are known. One critical step of the algorithm is the selection of a set of mutually non-interfering nodes at each step of the movement. These nodes will then progress to their desired destination and several such steps are taken until all nodes arrive at their desired positions. In [1] the authors present a heuristic that builds a feasible solution for the problem, but express no optimization concerns regarding the selection of the maximum number of nodes. In [2] an optimization method is indeed presented, but it is concerned with the derivation of the actual physical movements of the nodes and not node selection. We believe that it is possible to build improved paths if more care is taken when making such selection. In this paper, we attempt to develop a mechanism for creating an improved solution to the problem by using the capabilities of a genetic algorithm.

In the next section, we describe the base algorithm for adjusting node movements while preserving connectivity. We then follow by describing genetic algorithms and how they were applied in this specific case. We present some results and respective analysis, and then proceed to the conclusion of our work

2 Node Movement Coordination

In [1] a method for adjusting the movements of nodes and preserving global connectivity is presented. We will provide a brief description of some aspects of that work that are of relevance to ours.

Given an initial setup of globally connected nodes (ie. all nodes are able to communicate with each other, even though multi-hops may be required) and the desired final (also connected) setup, a set of waypoints is determined for each node.

The paths taken by the nodes are then determined by the sequence of waypoints by which each node should pass on its way to the destination. These waypoints are computed so that connectivity is always ensured all along the complete movement from the initial to the final positions. This is illustrated in Figure 1and Figure 2. In this example, if each node moved along the straight path between its current and final position (i.e. along the dashed line), node *C* would clearly loose connectivity with other nodes along the way. The same couldn't be said about nodes *A* and *B*, since these would always be within the specified transmission range of each other. To solve this problem, a set of paths is generated that ensures that, when node *C* loses connectivity with node *B* (the node it is initially connected to), it is already within transmission range of *A* and will remain so until both arrive at their new location.

The same number of waypoints is used for all the nodes, and all nodes should pass by each. All nodes are required to pass by their i-th waypoint simultaneously, and then proceed straight to the (i+1)-th waypoint with constant speed. This implies that each node will be required to move with speed proportional to the distance between the waypoints he is currently between.

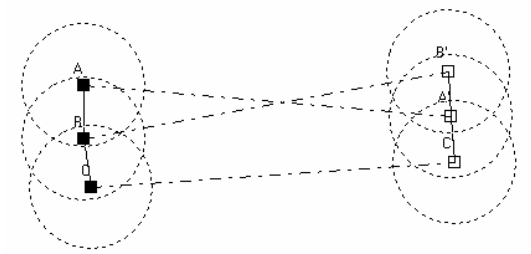

Figure 1 - Node C will loose connectivity when all nodes move from their initial positions

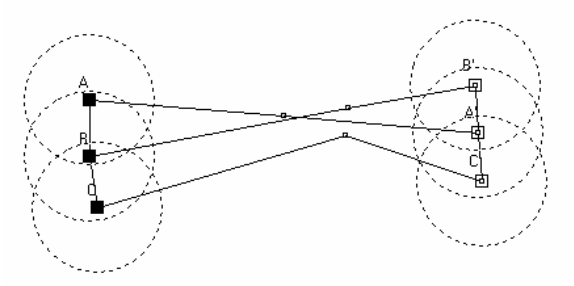

Figure 2 - Modified paths ensure connectivity all along the movement

The solution is constructed by handling a tree representation of the connectivity of initial and final topologies of the nodes - the *connectivity trees*. These trees are built by removing loops from a graph that represents the connections between nodes - the *connectivity graph*. It is shown in [1] that movement between the initial and final configuration can be accurately represented by operations that, when applied successively to the initial connectivity tree, will transform it into the final connectivity tree. These operations correspond in fact to specific movements on the geometric plane (specific waypoint positions) that are ensured to preserve the connectivity of the nodes.

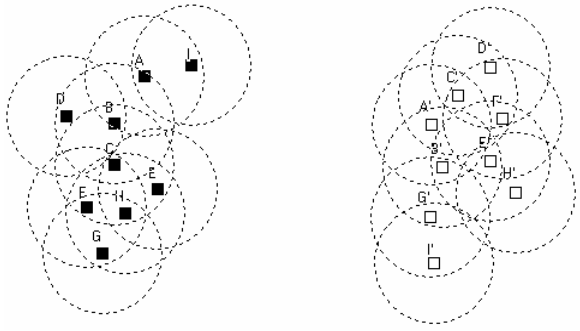

Figure 3 - Example of initial and final configuration of nodes.

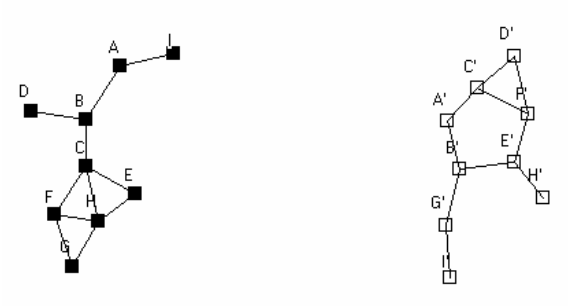

Figure 4 - Connectivity graphs corresponding to the example of Error! Reference source not found.**.**

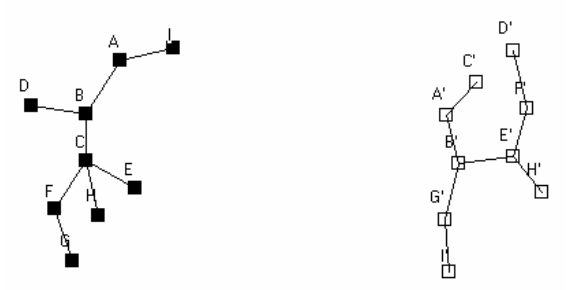

Figure 5 – Connectivity trees, after loop elimination is applied to graphs in Error! Reference source not found.**.**

In [1] it is established that given any node topology whose connectivity tree is equivalent to the desired final connectivity tree, it is trivial to find a path for each node that ensures connectivity along the way and that will place all nodes in the desired positions. The required movement is simply a linear motion from the current position to the desired position, setting the speed for each node in such a way that all will end movement simultaneously. See the following example (Figure 6):

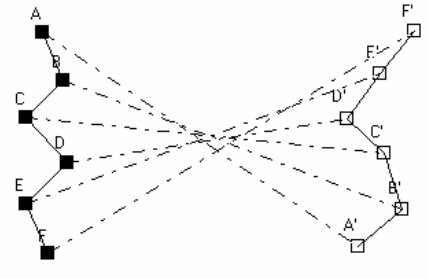

Figure 6 - Moving nodes from a connection tree equivalent to the desired configuration.

The importance of this fact is that it allows us to reduce the problem to finding the transformations required to convert the initial topology into one whose connectivity tree is equivalent to that of the desired final positioning, regardless of the actual physical positions of the nodes.

A possible first step in such transformation would be to (for instance) to make the following transformation to the initial connectivity tree:

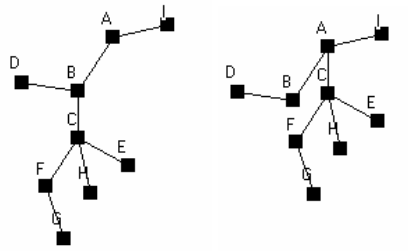

Figure 7 – Moving node *C* (and its sub-tree) to node *A*.

In this case we are moving the subtree with root C (we consider the root node to be A, for illustration purposes) to become a direct descendant of node A. This corresponds to the relative positions of A and C in the final connectivity tree. By applying such a process repeatedly to each node in (ie. moving it to (or towards) its parent node on the final connectivity tree), we can effectively derive a series of steps that describe the movements that the actual nodes need to make.

The movements required to place a node as descendant of the correct node are determined by finding the shortest path between both nodes. Applying this to the example of **Error! Reference source not found.**, this would generate the following candidate movements and paths:

C→B→A (meaning: move node C from node B to node A)

D→B→C→F (meaning: move node D from node B to node C, and then move node D from C to F)

E→C→B

F→C→E

G→F→C→B

H→C→E

I→A→B→C→F→G

It turns out that it is impossible to carry out all these movements simultaneously, so the movement to the final destination is made in several steps, with possibly different nodes moving in each step. For example, it isn't possible to ensure connectivity if movements E→C→B and F→C→E are taken simultaneously. This is elaborated on [1] but essentially, it can be synthesized by saying that if a node is moving to became descendant of other node that other node should not move.

In order to determine which nodes will actually move in each situation, a simple heuristic is employed in [1]. This heuristic randomly removes nodes from consideration until the movement of all remaining nodes is possible, and then tries to randomly add removed nodes while an unfeasible solution isn't created.
In this work we propose to use a genetic algorithm for making such a selection, with the goal of increasing the number of nodes moving in each step and thus reducing the number of steps required to attain the desired position.

3 Genetic Algorithms

We propose using genetic algorithms [6, 10] in order to find the optimal translation path from the initial to final configuration.
Genetic algorithms (GA) are a group of stochastic optimization techniques. These algorithms work with a set of candidate solutions to the problem (a population of individuals, using GA terminology) and seek to evolve them using concepts derived from genetics and natural selection. Each individual holds enough information (the *genes*) to describe a possible solution to the problem. They are evaluated regarding the quality of that solution (i.e. their *fitness* is computed), and a probabilistic selection method (based on the fitness of each individual) is used to find group of individuals (the *parents*) that will be used to create the next *generation* of the population. The individuals of the new generation are created by applying genetic inspired transformations (*operators*) to the parents. Among these transformations we can find *mutations* and *crossover*. The mutation operator does random changes to the genes, while the crossover combines parts of the genes of two parents to create a single individual. After multiple iterations, the quality of the population will increase, and when a predetermined stopping condition is met, the solution for the problem will be found on the genes of the best individual of the last generation.

1.Randomly initialize population
2.**While** stopping condition is not met
  a) Evaluate population
  b) Select parents
  c) Crossover
  d) Mutation
  e) Substitute old population

Algorithm 1 – Simple Genetic Algorithm.

There are, of course, multiple variants of this simple framework.

4 Genetic Algorithm Setup

Each solution is easily represented by a binary string, in which each position corresponds to whether or not each one of the possible movements is to be made in this step.
Standard one-point crossover and mutation are used to generate the descendants of the progenitors, whose selection is probabilistic based on a tournament scheme. Elitism is also implemented by carrying over the best individual in each population to the next one. The fitness of each individual is equal to the number of nodes that are specified to move, plus a significant bonus if the solution is valid. Invalid solutions are permitted, although they are heavily penalized.
We made the following modification to the standard genetic algorithm. Whenever the fitness of the best individual remains constant for a given number of iterations, a new feasible individual is created with the heuristic described in [1] and injected into the population. This serves two purposes:

- It ensures that a valid solution is found by the algorithm.
- It injects diversity to broaden the scope of the search when stuck at a (local) maximum.

The first point is important, because, since solutions are heavily constrained (as is the case with many engineering problems), it may be hard for the GA to find one initial good solution if the number of alternate paths is large. This givens the algorithm a push in the right direction, making sure a feasible solution is found (even if it is sub-optimal). One potential problem is that is that the first feasible solution will dominate the entire population, leading to poor diversity and low chances of finding better solutions, even if they are available. This effect is partially countered by the injection of new feasible individuals, greatly increasing the possibility of generating new and better solutions by application of the standard GA operators.
The genetic algorithm is a computation-intensive process. Presently, we propose that the application of such method should follow the guidelines indicated in [1], in which a more powerful node could compute all the waypoints and then send these to every other node.

The following picture shows the paths generated for a 27 node example.

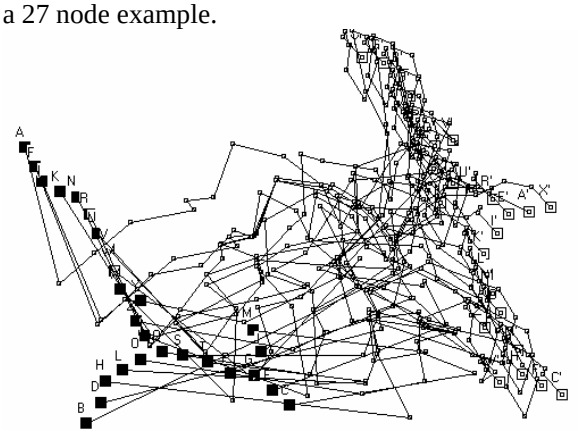

Figure 8 – Paths obtained by applying the genetic algorithm to node selection to a set of 27 nodes.

5 Results and Analysis

We conduct a series of experiements with randomly initialized nodes and target positions (the node sets had between 30 and 60 nodes). The genetic algorithm was configured to have a population of 50 solutions and ran for 1000 generations. Mutation and crossover were applied with probabilities 4% and 90%, respectively. New individuals were injected after 100 generations of constant fitness of the best individual.

The results are summarized in the tables below. Values are presented as the ratio between the number of steps required to move nodes to target position with or without optimization. At each run, a set of random nodes was generated and both the optimized and non-optimized algorithms were run. After 30 runs, the results averaged to the following.

Number of Nodes	Average Opt. /Non. Opt
30 node set	0,83
40 node set	0,82
50 node set	0,80
60 node set	0,79

It can be seen, from the table above, that the proposed approach does indeed offer effective improvements over the quality of the solutions generated by the unoptimized version.

Growing improvements are obtained consistently all over the runs. As the number of nodes increases, so does the magnitude of the difference between both approaches, albeit slightly. The simple heuristic may be unable to keep up the increased complexity that emerges when the node number is increased.

In light of the results, the approach chosen in this work seems to be adequate for the problem. The changes to the standard GA seemed to be adequate for the task at hand.

6 Conclusions and Future Work

In this work we presented an improvement over an existing algorithm for coordinating the movements of mobile nodes in a ad-hoc network environment, with the purpose of ensuring no partitioning of the network ever occurs. The changes introduced in the original work offer significant improvement over the quality of the original solutions, while ensure that their connection-preserving properties of the maintaining. The increased performance results from an increase in the concurrence of node movements.

Other works address different optimization issues in this area, namely physical movement optimization It would be interesting in the future to combine both approaches into an holistic optimization scheme, where simultaneous optimization at different levels would be conducted, with global evaluation of the impact.

In order to enhance the applicability and scalability of the algorithm, and considering the environment the algorithm is to be executed in, thought should be given to deriving distributed mechanisms to accomplish the same tasks.

References

[1] "Choosing Paths that Prevent Network Partitioning in Mobile Ad-hoc Networks", F. Coutinho, J. Barreiros, J. Fonseca, *Presented at WFCS 2004, Vienna, Austria, Sept. 2004*

[2] "Mobile Node Path Optimization with Network Partitioning Avoidance", F. Coutinho, J. Barreiros, submitted *to CEE 2005, Coimbra, Portugal.*

[3] "A Comparison of Routing Strategies for Vehicular Ad Hoc Networks", H. Füßler, M. Mauve, H. Hartenstein, M. Käsemann, D. Vollmer, *Fakultät für Mathematik und Informatik, Universität Mannheim, Reihe Informatik, Germany*, 2002.

[4] "Wireless, mobile ad-hoc networking", M. Gerla, G. Pei, and S.-J. Lee, *Presented at IEEE/ACM FOCUS'99, New Brunswick, NJ*, May 1999.

[5] "Position based routing algorithms for ad-hoc networks: a taxonomy", S. Giordano, I. Stojmenovic, L. Blazevic, *Institute for Computer Communications and Applications, École Polytechnique Fédérale de Lausanne, Switzerland*, 2001.

[6] "Genetic Algorithms in Search, optimization and machine learning", Goldberg, D. Addison Wesley, 1989.

[7] "The handbook of ad hoc wireless networks", Mohammad Ilyas (Ed.), CRC Press, 2003, ISBN 0-8493-1332-5.

[8] "A review of current routing protocols for ad hoc mobile wireless networks", E. Royer, C.-K. Toh, *IEEE Personal Communications*, April 1999.

[9] "A Survey on position-based routing in mobile ad-hoc networks", M. Mauve, J. Widmer, H. Hartenstein, *Praktische Informatik IV, Universität Mannheim, Germany*, 2001.

[10] "An introduction to Genetic Algorithms", Mitchel, M., MIT Press, 1996

[11] "Ad hoc mobile wireless networks : protocols and systems", C.-K. Toh, Prentice Hall, 2002, ISBN 0-13-007817-4.

[12] "Ad hoc networking", Charles E. Perkins (Ed.), Addison-Wesley, 2001, ISBN 0-201-30976-9.