Connectivity Maintenance of a Set of Agents through MST-based Algorithm

Cézanne Alves
Computer Science Department
Federal University of Tocantins
Palmas/Tocantins - Brazil

Warley Gramacho da Silva
Computer Science Department
Federal University of Tocantins
Palmas/Tocantins - Brazil

Rafael Lima de Carvalho
Computer Science Department
Federal University of Tocantins
Palmas/Tocantins - Brazil

ABSTRACT
In this paper, it is proposed a solution to the problem of positioning a set of agents that play the role of pursing a set of moving targets, while the global connectivity among such agents is maintained throughout positioning a second set of relay agents. The role of the agents consists of organizing themselves in order to allow the underlying network to stretch at its maximum (maximizing the action of pursuers), while ensuring the connectivity. In order to do that, this work proposes a positioning algorithm that uses the Minimum Spanning Tree (MST) in a way that maximizes the mobility of the nodes while deciding on the position of relays and pursuers. The approach is validated through experimental simulations using a set of behaviors to some deployed targets showing the feasibility of the proposed solution.

General Terms
Connectivity Problem, Pursuer-evasion, Dynamic positioning problem.

Keywords
Connectivity Problem, Pursuer-evasion, Minimum Spanning Tree, Dynamic positioning problem.

1. INTRODUCTION
In disaster scenarios, dynamic connectivity among search agents may be of vital importance to the success of a rescue mission. This scenario can be modelled as the problem of given a set of possible moving targets, a set of pursuers for which the mission is to reach the set of targets, and an additional set of relay agents, the problem asks for the positioning of the latter set in order to maximize the action of the pursuers and ensure global connectivity. Figure 1 shows the considered scenario.

In [1], the authors considered the challenges of dealing with a simple situation: given two static pursuers and a set of relays deployed around them, how to move the minimum number of relays with minimal effort in order to establish connectivity between the static pursuers. Despite the simplicity of the scenario, the authors showed that in order to find the optimal configuration of movements, there should be a known ordering of such relays. Furthermore, in [2], it is proposed an approach to the problem of computing the minimum number of robotic routers (and their motion strategies) in order to maintain the connectivity of a single pursuer to a base station. It was considered the case where the polygon is concave in order to calculate optimally the positioning based on the velocity vector acting over each agent.

Kim *et.al.* [3] proposed an optimization model based on the weighted Laplacian matrix of the underlying graph induced by the positions of the agents. As long as the second smallest eigenvalue of the Laplacian matrix remains greater than zero, the network graph remains connected [4]. With such information, they proposed a solution using an iterative algorithm that optimizes the control of each agent, to perform missions such as rendezvous without losing connectivity. Furthermore, in [5] the authors reached the same objectives as [3] but in a distributed way, i.e., the agent needed only to know about its local neighborhood in order to decide the next position and still ensuring global connectivity.

In addition, in [6] the authors showed a mixed integer linear programming approach for a pursuit-evasion which included optional connectivity constraints. They considered deciding over an occupancy grid for which the set of pursuers are trying to cover a given area. The discretized area included obstacles and the cells were labeled over a discrete time horizon.

In [7], the authors presented an extension to the Darwinian Particle Swarm Optimization (DPSO) algorithm, which they named Robot DPSO (RDPSO). The RDPSO main three key aspects are: a) ensuring network connectivity; b) social exclusion and inclusion, and c) obstacle avoidance. The Darwinian PSO [8] is an extended version of the traditional PSO algorithm in a way that natural selection or survival-of-the-fittest is added in order to enhance the chances for escaping from local optima.

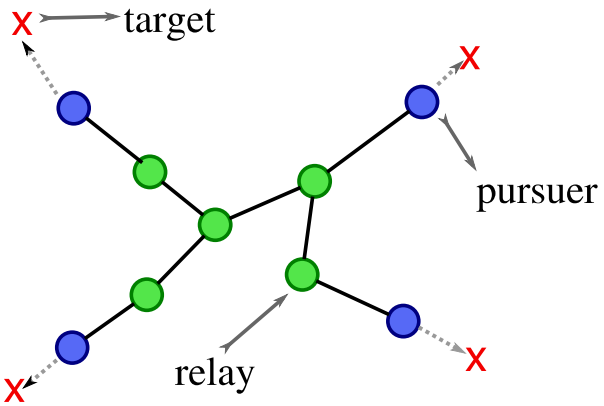

Fig 1: General view of the pursuer-evasion problem under connectivity constraints.

Despite the existence of many approaches for this problem, in this paper it is proposed a quadratic model for which the positioning of the relays is based on a minimum spanning tree (MST) approach. MST is a fundamental algorithm found in the graph theory and as shown by this work, it can be used to

drive a set of pursuers and relays in order to maintain them connected, while the pursuers still fulfill their mission.

The general structure of this paper is as follows: Section 2 shows the fundamental theories as well as the mathematical model of the problem. Section 3 describes the proposed approach to the problem. In addition, Section 4 presents the computational simulations showing the feasibility of the proposed solution. Lastly, Section 5 points the general discussion about the results and points some future directions to this research.

2. THEORETICAL REVIEW

Let $P = \{p_1, p_2, \dots, p_n\}$ be a set of agents whose objective is to move towards their respective targets of the set $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. Let $R = \{r_1, r_2, \dots, r_m\}$ be a set of relay agents whose objective is to position themselves in order to provide connectivity to the elements of P. Two elements of x_1 and $x_2 \in P \cup R$ can communicate only if $\|x_1 - x_2\| \leq D$, the maximum communication distance. In order to describe the movement of the agents, and better represent the limitations of a real-time implementation, the time is discrete, i.e., from instant t to instant $t+1$, with interval of Δt time, each agent must decide on its new position based on the distance it can travel in that time interval and the connectivity constraints. This problem can be defined as a quadratic programming model for which it is desired to minimize the average distance among the elements of P and their counterparts of Θ, subject to the connectivity restriction of the proximity graph induced by the position of the elements of $P \cup R$.

To denote time, let $P(t) = \{p_1(t), p_2(t), \dots, p_n(t)\}$ be the positions of the pursuers at time t. For this work, it is assumed that $p_i(t) \in \mathbb{R}^2$. The sets $\Theta(t)$ and $R(t)$ are defined in a similar way. In this way, the quadratic model of the aforementioned problem is given by:

$$\min_{P(t+1)\cup R(t+1)\in\{\{x_1,x_2,\dots,x_{n+m}\}|x_i\in\mathbb{R}^2\}} \frac{1}{n}\sum_{i}^{n} \|p_i(t+1) - \theta_i(t)\| \#(1)$$

$$s.t.$$

$$\left(x(t) - x(t+1)\right)^2 \leq D_{step}^2 \forall x \in P \cup R \#(2)$$

$$G(R(t+1) \cup P(t+1)) \; is \; connected \#(3)$$

Where $G(X)$ is the proximity graph induced by the position in $\mathbb{R}^2$of the elements of the set X, and D_{step} is the maximum distance that any agent can move from instant t to $t+1$

The problem asks for solutions to move the relays in order to provide connectivity and mobility to the pursuers in the directions of their targets, and how to move the pursuers given the restriction of the positions of the relays. However, the restrictions imply as well in finding which edges should be broken in order to increase the reachability of the objectives, while still maintaining global connectivity.

Let us define the restriction topology as the set of edges of the network whose integrity we decide to maintain in order to assure connectivity.

As the hosts move independently and have limited information about their neighbors, in order to achieve the objective (1), we want to avoid restricting the movement of some node while the graph is still able to stretch. This approach gives rise to the following problems:

Problem 1: Given the current node positions $V(t)$ of a network $G(t) = (V(t), E(t))$, which restriction topology $E' \subseteq E(t)$ maximizes the mobility of the nodes without disconnecting the network?

Problem 2: Given the restriction topology $E' \subseteq E$, where to move the relays in order to maximize the mobility of the nodes?

Problem 3: How each node deals with the uncertainty of the movement of their neighbors in order to assure connectivity?

The next section presents the proposed the connectivity maintenance solution, with detailed discussions of the above sub-problems in subsections 3.3, 3.4, and 3.5, respectively.

3. MST-BASED APPROACH

The proposed method consists in keeping the network connected by the edges of its Minimum Spanning Tree (MST), ignoring completely the others edges and letting them break freely, this way we can substitute the restriction 3 by:

$$\forall (i,j) \in T^*: \left(x_i - x_j\right)^2 \leq D^2 \#(4)$$

where E is the set of edges of the MST.

The relays, then, move to the middle of their neighbors with the intent of increasing the network reach.

The MST is recomputed in each iteration to guarantee optimality in case the movements of the hosts make the previous tree not minimal. The pursuers just move in the direction of theirs targets, with the only restriction of not breaking any MST edge, and the relays move to middle of their neighbors in the MST to increase the reach of the network, as if the edges exerted traction on them. Figure 2 illustrates this procedure.

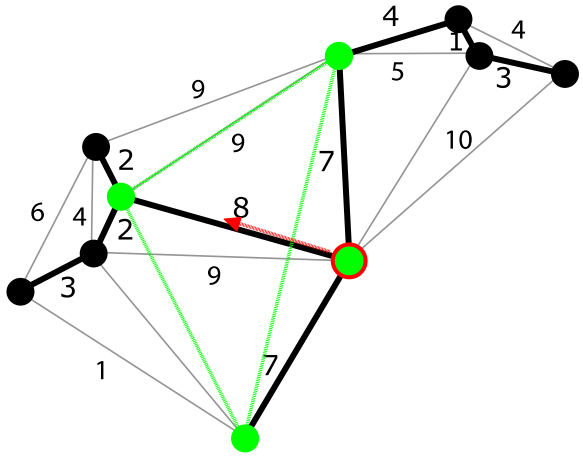

Fig 2: Movement of the relay – the highlighted relay moves to the middle of its neighbors, increasing their mobility.

3.1 Problem Variations

We defined the problem as it is, with pursuers and relays, to simplify discussion and focus on the more relevant aspects. However, the proposed solution also applies to more broader scenarios.

There could be static nodes representing fixed base stations, and they would be treated similarly as the pursuers, or nodes that transition between the state of pursuing, static and

relaying, and the relays would move to increase the mobility of the network.

In this work when we refer as **relaying node** or **relay** to any node that in a given moment has the sole purpose of routing packages and move only-and-always to maintain connectivity and provide mobility to other nodes. To all the other types of nodes, we may refer as **active nodes** or **pursuers**. To nodes that can transit between these states we refer as **auto-nodes.**

3.1.1 Routing pursuers

Another variation the problem may have is: whether or not active nodes are allowed to route network packages.

If there are active nodes not allowed to route packages, we compute a connected subgraph with only the routing nodes. Then, to form the restriction subgraph, we connect each non-routing actives to its closest routing node so as to insure non-routing actives always have degree 1. In this case, we call the immediate relay of an active node the **relay provider**.

However, regardless of being able to route packages, there is a problem in allowing actives as branch nodes in the restriction tree: As, in principle, the relays rely only on the position of the neighbors to decide their movement, actives can be trapped in a branch of the network, as illustrated by Figure 3.

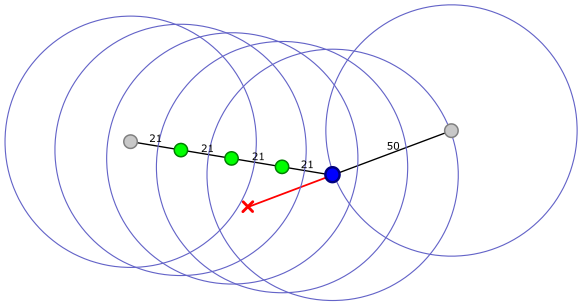

Fig 3: Pursuer trapped in the MST. The relays (green) could link the two static nodes (gray) directly, allowing the pursuer (blue) to reach the target.

Our strategy to overcome this is to always make the active nodes as leaves of the restriction tree, even when it's not a routing requirement. This imposes some restrictions in scenarios where there is auto-nodes, where is preferable to activate auto-nodes in leaves, as actives in branches could become trapped.

3.2 Computation of the MST

The simplest method to compute the MST is to elect one node as the leader, which will compute the MST every round. As the network is connected, every node informs its position to the leader through network messages in the beginning of the round, and after computing the MST, the leader informs the edges that the hosts must not break. In a round where the MST changes, the leader first request the conservation of the new links. Only after acknowledgement of the request by all involved hosts, with confirmation of the integrity of the links, the leader informs which links may now be broken. Here we use the Kruskal Algorithm [9].

As the connectivity doesn't rely on the MST being up-to-date, but in it being the same for every host, the MST doesn't need to be recomputed every round. This could allow less overhead in network traffic and higher frequency in updating the relay function, making the relays more responsive to movements of the neighbors.

3.3 MST as Restriction Subgraph

At first sight, one could think that using the MST to restrict the movement would result in a network with small reach, since it has smaller edges than other subgraphs. However, we do not use the MST to define the position of the nodes, but only to choose which edges to avoid breaking. In addition, the limit of distance of a node to its neighbor is not the edge between them, but D, and smaller edges allow for a greater stretch until reaching this limit.

Let's define the mobility of a node x, $l_x = \min_{(x,j)\in E} D - \|x - j\|$, as the greater distance it can move in any direction without breaking an edge assuming the others remain still. And the mobility of the network G, $l_G = \min_{x\in V} l_x$, as the greater distance any node can move in any direction without breaking an edge.

First, let's prove that a non-tree connected spanning subgraph is redundant as a restriction subgraph, and that there is always an optimum spanning tree to be used as the restriction subgraph.

Let $G' = (V, E')$ be a non-tree connected spanning subgraph of G, that is, G' has cycles. G' has a node x that belongs to a cycle, therefore x is connected to other nodes by two (or more) paths that start from itself by distinct edges. If we opt by maintain the edges of E', x would have its movement limited by the neighbors of this two paths, that is, $l_x = \min_{(x,j)\in E'} D - \|x - j\|$.

As these edges belongs to a cycle, one of them, say (x, y), may be broke without disconnecting the graph, which may increase mobility to x. That is, by removing (x, y) of the restriction subgraph, the mobility of x becomes:

$$l_x = \min_{(x,j)\in E'-\{(x,y)\}}(D - \|x - j\|) \leq \min_{(x,j)\in E'}(D - \|x - j\|) \#\blacksquare (5)$$

3.3.1 Optimality of the MST

Given a connected spanning subgraph $G' = (V, E')$, it's easy to see that $l_{G'} = D - w_{e'}$, where e' is the longest edge of E'.

The reason for choosing the MST to restrict connectivity is that it minimizes the longest edge in the network, that is, it maximizes the network's mobility.

Proof: Suppose a MST of a network G, $T = (V, A)$, whose longest edge is a, and a non-minimum spanning tree of G, $T' = (V, A')$, whose the longest edge a' is smaller than a. By removing a from T, we get two connected components that induce a 2-partition of V. Any spanning tree of G has one and only one edge connecting some node of one set of any 2-partition of V to the other. So, let e' the edge of t that connects the sets induced by removing a. By supposition $e' < a$, and we could insert e' in T making it connected again but with smaller weight, that is, T was not minimum, a contradiction.

Therefore T minimizes the longest edge between all the connected spanning subgraphs of G, and hence is a spanning connected subgraph that maximizes its mobility■.

3.4 Moving the relays

To simplify the discussion, this section explains the movement of a relay r as if its neighbors remained still throughout the rounds. We also assume the restriction tree always maintain active nodes as leaves.

In time t, r compute its ideal final position r' and spend the next round moving at full speed to the desired position until it reaches it or the round ends. So,

$$r(t+1) = \begin{cases} r(t) + \widehat{(r' - r(t))} \times D_{step}, \text{ if } (r' - r(t)) > D_{step} \\ r', \text{ otherwise} \end{cases} \#(6)$$

To ensure all the nodes of the network have mobility, we wish to, in each iteration, move the relays so as to minimize the distance to their neighbors. Thus, we need to choose a function upon the distance of the neighbors to minimize.

3.4.1 Desired Properties

Assuming stationary neighbors, it is desirable to compute the final desired position in the first round, regardless of the actual position of r in the current round, so the relay can move at full speed to this final position, restricted only by its physical limits of speed, acceleration and deceleration. This would result in faster response of the relays to the movement of the active nodes

3.4.2 Relay Objective Function

Since the longest edges are more restrictive, we wish to give more weight to the farthest neighbors. Let $\aleph_x$ be the set of neighbors of the node x in the restriction tree, and x' be the next position we want to move x to.

The chosen function was:

$$\forall r \in R, \qquad r' = \arg\min_{x \in \mathbb{R}^2} \left(\sum_{y \in \aleph_{r(t)}} \|x - y\|^2 \right) \#(7)$$

Which, if unrestricted by (3), results in

$$\frac{1}{n} \sum_{x \in \aleph_r} (x) \#(8)$$

This simple function is easily minimized algebraically, and the computation of the minimum takes $O(\aleph_r)$ time, and does not varies with the current position of x, resulting in the final position been known in the first round (assuming the neighbors don't move). This way we can compute the optimal position for the relay very fast in only one round, even for dense graphs, since we only consider the neighbors on the restriction tree.

The procedure to guaranteeing a solution inside the restriction (3) is described below.

3.5 Preventing Disconnection

To ensure connectivity, we let the nodes free to move at their maximum velocity to the next desired position, unless they are able to break one edge until the next round with their maximum velocity. Since one node does not have information about the next movement of its neighbors, it has to prevent for the worst movement possible. For the node n_1 and one neighbor n_2, we split this restriction between both: The maximum distance n_1 can move to a given direction, is half the remaining distance to reaching the maximum radius in such direction, so n_2 can move the same distance in the opposite direction.

We achieve this by pruning the movement vector of n_1 with a circle of radius $\frac{R}{2}$ centered in the middle of n_1 and n_2. It is easy to see the correctness of this procedure: Since the circle is the same for both, both can move to whatever position inside the circle and $\|n_1(t+1) - n_2(t+1)\| \leq D$.

Let C_{nj} be the disc centered in $\frac{n+j}{2}$ with radius $\frac{R}{2}$, and n' be the objective of the node n if it's a pursuer, or the position obtained by the relay moving function, if it's a relay.

Let $B_n = \bigcap_{j \in \aleph_n} C_{nj}$. Since n lies inside B_n,

$$\exists x \in \overline{nn'} | x \in B_n \#(9)$$

We define then $n'_B = \arg\min_{x \in \overline{nn'} \cap B_n} \|x - n'\|$, and substitute the restriction (2) by:

$$\forall n \in P(t) \cup R(t): n(t+1) = \begin{cases} n(t) + \widehat{(n'_B - n(t))} \times D_{step}, \text{ if } (n'_B - n(t)) > D_{step} \\ n'_B, \text{ otherwise} \end{cases} \#(10)$$

This can be done in $O(\aleph_n)$, by pruning the line segment $\overline{nn'}$ with $C_{nj} \forall j \in \aleph_n$

Such procedure can slow down the progress of n only when $D_{step} > \frac{D - \|n-j\|}{2}$, for some $j \in \aleph_n$, which, for small values of Δ_t, will happen only when the edge (n, j) is near its maximum stretch.

4. COMPUTATIONAL EXPERIMENTS

To test the efficacy of the technic we ran the algorithm in the test scenarios described in [10], with simple heuristics for assignment of the targets. As in that work, we used the Hungarian method [11] to assign pursuers to targets, which minimizes the sum of the distances from the pursuers to their respective targets. Although this method is not ideal for instances where the targets are out of the network's reach and the swarm need to move itself entirely to reach an objective, these scenarios can give good evidence on how the technic performs when it is possible to reach the targets by changing the topology of the network.

4.1 Out-of-reach Targets

To avoid the network of being stuck when the targets are out of reach, we implemented two counter-measures:

4.1.1 Swarm direction

When some pursuer has its velocity restricted by the stretch of the network, we make all the pursuers headed in an opposite direction of the swarm to retreat, that is, move toward its relay provider.

We defined the swarm direction as:

$$\overrightarrow{Sd} = \sum_{p_i \in P} \frac{1}{\|p_i - \theta_i\|} \widehat{p_i \theta_i} \#(11)$$

So, when $\|\overrightarrow{Sd}\| > 0$ and

$$\exists p_i \in P : \|p_i(t) - p_i(t-1)\| < D_{step} \wedge \overrightarrow{Sd} \cdot \overrightarrow{p_i \theta_i} > 0 \#(12)$$

We make all p_i such that $\overrightarrow{Sd} \cdot \overrightarrow{p_i \theta_i} \leq 0$ retreat.

4.1.2 Inactive pursuers

When the above measure fails to insure progress, a more aggressive measure is taken, which can deactivate even nodes aligned with the direction of the swarm. Let's partition P into two always disjoint sets P_a, the set of active pursuers, and P_d, the set of inactive pursuers, which always retreat. Initially, $P_d = \emptyset$ and all pursuers start retreating when put in this set.

If during $|R| + 1$ consecutive rounds

$$\nexists p_i \in P_a : \|p_i(t) - p_i(t-1)\| > \frac{D_{step}}{2^{|R|+1}} \#(13)$$

and $P_a > 1$, then:

$$P_d(t+1) = P_d(t) + \{\arg\max_{p_i \in P_a} \|p_i - \theta_i\|\} \#(14)$$

When any active pursuer changes its target, $P_a = P$

4.2 Test Scenarios and Results

The benchmarking scenarios presented in [10] consist of two arbitrary placements of targets in $\mathbb{R}^2$ with different behaviors displayed by the targets. **Table 1** presents the targets' positions of the scenarios, and **Table 2** describes the behavior schemes.

Table 1. Targets locations

Point	AllLeft	Symmetric
1	$(-60, 35)$	$(0, 60)$
2	$(-65, 25)$	$(0, -60)$
3	$(-50, 10)$	$(-60, 0)$
4	$(-50, -8)$	$(60, 0)$
5	$(-65, -12)$	$(-75, -30)$
6	$(-60, -15)$	$(-75, 30)$
7	$(-75, -20)$	$(75, 30)$
8	-	$(75, -30)$

Table 2. Behavior schemes description

Scheme	Behavior
Escape	The targets move at the direction opposite to their closest pursuer
Static	The targets remain still during all simulation
Spiral	The targets follow a discretized counter-clockwise logarithmic spiral $r(\phi) = ae^{b\phi}$centered in their initial position, with $a = 1$, $b = 0.1$, and ϕ sampled at an interval of $1\ rad$ starting at $1\ rad$. If a target reaches a point of the spiral, it stops, sample the next point, and head to it in the next round.
Cooperative	The targets move towards their closest pursuer
Random	At each round, the targets pick a random direction and velocity.

In the benchmark, $D = 15$, the maximum velocities were $P_{maxV} = 5$, $R_{maxV} = 5$, $\Theta_{maxV} = 2$, and every target, except those with random behavior, always move at their maximum velocity.

When a pursuer has its target in an effective range of 5 units, it captures the target and is assigned to another.

All nodes are deployed at position $(0,0)$ and the simulation ends when the last target is captured, or when 200 simulated second have passed.

We simulated with a $\Delta_t = 0.01s$, which was computed faster than real-time.

The results are presented in **Figure 4**.

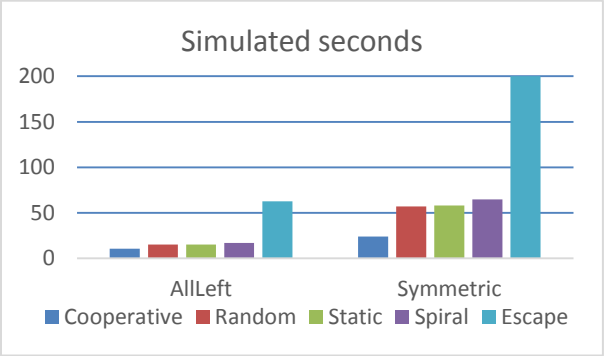

Fig 4: Simulation results

The Symmetric scenario has targets in opposite direction too distant for the pursuers to be able to reach to their assigned targets at the same time, requiring the use of some counter-measure.

The static and escape instances of the symmetric scenario made the first counter-measure impossible: During initial spreading, all pursuers remain at symmetrically opposing direction and with same distance of their targets, making $\|\overrightarrow{Sd}\| = 0$, this caused the pursuers stop for a brief moment until the more aggressive counter-measure took place.

Only the Symmetric-escape instance was not completed within 200 seconds, with only 3 targets captured of a total of 8. This is due to the increasing distance between the targets: From the moment the targets become distant form each other more than the maximum diameter of the network, it becomes impossible to pursue more than one target at a time, and the relay positioning and topology control become less relevant as the distance increases, whereas the assignment order becomes crucial.

In the AllLeft scenario – with exception of the escape instance, where opposing targets reach distance greater than the maximum diameter of the network – the relays provided uncompromised connectivity to the pursuers, that is, all the pursuers moved at maximum speed toward their targets during the entire simulation.

5. CONCLUSION

In this work, it has been presented an approach that deals with the connectivity problem in a scenario of pursuit-evasion. In a simplified version of an MST-based approach, the developed protocol could ensure global connectivity while allowing the pursuers to reach their objectives. In the literature, there are many different approaches which uses a discretized version of the area, or more complex approaches such as using the Laplacian matrices and eigenvalues decomposition and optimization.

Experiments confirmed that the solution is able to change topology under demand of the pursuers. There was significant loss of performance only when the targets were beyond simultaneous reach of the swarm.

A future appointment to overcome this problem is the development of an integrated assignment protocol to enable the swarm to receive higher level tasks and decide a near optimum order for completion.

6. REFERENCES

[1] Tekdas, O., Kumar, Y., Isler, V. *et al*. 2012. Building a Communication Bridge with Mobile Hubs. In IEEE T. Automation Science and Engineering v. 9, n. 1, pp. 171–176.

[2] Tekdas, O., et al. 2010. Maintaining connectivity in environments with obstacles." In: IEEE ICRA, pp. 1952–1957.

[3] Kim, Y., Mesbahi, M. 2006. On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian, IEEE Transactions on Automatic Control, v. 51, n. 1, pp. 116–120.

[4] Godsil, C., Royle, G. 2001. Algebraic Graph Theory, v. 207, Graduate Texts in Mathematics. Volume 207 of Graduate Texts in Mathematics. Springer.

[5] De Gennaro, M., Jadbabaie, A. 2006. Decentralized Control of Connectivity for Multi-Agent Systems. In: 45th IEEE Conference on Decision and Control, pp. 3628–3633.

[6] Thunberg, J., Ogren, P. 2011. A Mixed Integer Linear Programming approach to pursuit evasion problems with optional connectivity constraints, Autonomous Robots, v. 31, n. 4, pp. 333–343.

[7] Tillet, J., Rao, T. M., Sahin, F. 2005. Darwinian Particle Swarm Optimization. In: 2nd Indian International Conference on Artificial Intelligence, pp. 1474–1487

[8] Couceiro, M., Rocha, R., and Ferreira, N. 2011. A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms. In IEEE International Symposium on Safety, Security, and Rescue Robotics, pp. 327.

[9] KRUSKAL, J. B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical society, JSTOR, v. 7, n. 1, pp. 48-50

[10] Carvalho, R. L. d. 2016. Multitarget Tracking System with Connectivity Constraints. Doctoral Thesis. Federal University of Rio de Janeiro, COPPE

[11] Kuhn, H. W., Yaw, B. 1955. The Hungarian method for the assignment problem, Naval Res. Logist. Quart, pp. 83-97.