Maintaining Ad-Hoc Communication Network in Area Protection Scenarios with Adversarial Agents

Marika Ivanová  
Department of Informatics  
Faculty of Mathematics and Natural Sciences  
University of Bergen, Norway  
marika.ivanova@uib.no

Pavel Surynek  
Faculty of Information Technology  
Czech Technical University  
Czech Republic  
pavel.surynek@fit.cvut.cz

Diep Thi Ngoc Nguyen  
AIRC, National Institute of Advanced Industrial Science and Technology (AIST)  
Japan  
diep.nguyen@aist.go.jp

Abstract

We address a problem of area protection in graph-based scenarios with multiple mobile agents where connectivity is maintained among agents to ensure they can communicate. The problem consists of two adversarial teams of agents that move in an undirected graph shared by both teams. Agents are placed in vertices of the graph; at most one agent can occupy a vertex; and they can move into adjacent vertices in a conflict free way. Teams have asymmetric goals: the aim of one team - attackers - is to invade into given area while the aim of the opponent team - defenders - is to protect the area from being entered by attackers by occupying selected vertices. The team of defenders need to maintain connectivity of vertices occupied by its own agents in a visibility graph. The visibility graph models possibility of communication between pairs of vertices.

We study strategies for allocating vertices to be occupied by the team of defenders to block attacking agents where connectivity is maintained at the same time. To do this we reserve a subset of defending agents that do not try to block the attackers but instead are placed to support connectivity of the team. The performance of strategies is tested in multiple benchmarks. The success of a strategy is heavily dependent on the type of the instance, and so one of the contributions of this work is that we identify suitable strategies for diverse instance types.

Introduction

In this work we study a generalization of Area Protection Problem (APP) with connectivity maintenance (APPC). APP is already a computationally hard problem (Ivanova, Surynek, and Hirayama 2018) (namely PSPACE-hard). In addition to APP, where two teams of mobile agents move in an undirected graph in a conflict free way, a possibility of communication among agents is required in APPC. APP itself can be regarded as a modification of known problem of Adversarial Cooperative Path Finding (ACPF) (Ivanová and Surynek 2014) where two teams of agents compete in reaching their target positions. Unlike ACPF, where the goals of teams of agents are symmetric - agents of each team try to reach their targets as first, the adversarial teams in APP have different objectives. The first team of attackers contains agents whose goal is to reach a pre-defined target location in the area being protected by the second team of defenders. Each attacker has a unique target in the protected area, and each target is assigned to exactly one attacker. The opponent team of defenders tries to prevent the attackers from reaching their targets by occupying selected locations, referred to as destinations, so that they cannot be passed by attackers. Specially in APPC, we require that vertices occupied by the defender team always form a connected subgraph with respect to the visibility graph. We assume that both teams use the same cooperative path-finding (CPF) algorithm for reaching temporarily selected locations.

Another distinction between ACPF and APP is a definition of victory of a team. A team in ACPF wins if all its agents reach their targets and agents of no other team manage to do so earlier. In APP, the team of defenders wins if all attackers are kept out of their targets. Our effort is to design destination allocation strategies for the defending team. A hard constraint that can never be violated will be that defending agents always form a connected component. Success of the strategy is measured from the defenders’ perspective via an objective function which plays a role of soft constraint (area protection may not be perfect). The following objective functions can be pursued:

1. maximize the number of target locations that are not captured by the corresponding attacker
2. maximize the number of targets that are not captured by the corresponding attacker within a given time limit
3. maximize the sum of distances between the attackers and their corresponding targets
4. minimize the time spent at captured targets

The common feature of APP, APPC and other related problems is that once a location is occupied by an agent it cannot be entered by another agent until it is first vacated by the agent which occupies it (opposing agent cannot push it out). This property represents a key tool for the defenders to protect the area.

APPC has many real-life motivations from the domains of access denial operations, robotics with adversarial teams of robots or other type of penetrators (Agmon, Kaminka, and Kraus 2011), and computer games. In most practical applications, agents of given team need to communicate with each other while individual robots can communicate at short distances.
visual range only as it has been already done in contemporary multi-robot systems. Hence it needs to be ensured that
the communication reaches every agent of the team. Such
property can be modeled as connectivity over the visibility
graph whose edges represent possibility of communication
between pairs of vertices.

Our contribution consists in suggesting several on-line
solving strategies for defenders that determine suitable ver-
tices to be occupied by defenders so that attacker agents
cannot pass into the protected area, and connectivity in the
defender team is maintained. We conduct an experimental
evaluation of the proposed strategies and assess their suitabil-
ity for diverse types of APPC instances.

Communication Maintenance
APPC requires, in addition to APP, that any two defenders are
able to communicate with each other at any time during their
movement. The communication within the team of defenders
is modeled by a connectivity of subgraph of the visibility
graph induced by the set of occupied vertices. The visibility
graph is derived from the graph in which agents move; it has
the same set of vertices, but the set of edges is different in
general. This assumption allows us to model the inability to
communicate of two agents, if there is an obstacle between
them.

In various practical applications of APP, the possibility
of sending messages among the agents is often demanded.
Agents may be equipped by an omnidirectional antenna or
visual communication device (such as LEDs and IR sensors
(Rubenstein et al. 2014)), and hence a message reaches all
nodes within the communication range of its sender. This
feature is often referred to as wireless advantage (Wieselthier,
Nguyen, and Ephremides 2002). We assume that the agents
have equal and constant communication range, and that they
can also work as transceivers, which means that they have
the ability to both transmit and receive a signal.

Related Work
APPC, APP, as well as ACPF share the way how movement
of agents is treated with the basic variant of cooperative path-
finding problem - CPF (multi-agent path-finding - MAPF)
(Silver 2005; Ryan 2008; Wang and Botea 2011). In CPF,
the task is to plan the movement of agents so that each agent
reaches its unique target in a conflict free manner. Move-
ments of agents in APPC at the low reactive level are assumed
to be planned by some CPF algorithm where agents of own
team cooperate while the opposing agents are considered as
obstacles.

There exist multiple CPF algorithms both complete and
incomplete as well as optimal and sub-optimal under various
objective functions. A good compromise between quality of
solutions and the speed of solving is represented by subop-
timal/incomplete search based methods which are derived from
the standard A* algorithm. These methods include
LRA*, CA*, HCA*, and WHCA* (Silver 2005). They pro-
vide solutions where individual paths of agents tend to be
close to respective shortest paths connecting agents’ loca-
tions and their targets. Conflict avoidance among agents is
implemented via a so called reservation table in case of CA*,
Consider an embedding of $G$ in a plane such that all edges have length 1 and each vertex $v \in V$ has coordinates $(x_v, y_v)$. The physical location $l_v$ represented by $v$ is the unit square area centered at $C_v = (x_v, y_v)$. Furthermore, let $O$ denote the set of square locations representing obstacles.

Let $r$ be the visibility range, i.e., the maximum distance between two locations such that two agents located at them can communicate together. The locations $l_u$ and $l_v$ can communicate with each other if the line segment $C_uC_v$ does not intersect any obstacle and the length of the shortest path $p_{uv}$ from $u$ to $v$ is at most $r$; shortly we say that $l_u$ is visible from $l_v$ and vice-versa. The visibility graph $G_r = (V, E_r)$ for a visibility range $r$ contains edges between every two vertices that are mutually visible, formally: $(u, v) \in E_r \iff C_uC_v \cap O = \emptyset \land |p_{uv}| \leq r$. For any $S \subseteq V$ we use $G_r[S]$ in order to denote a subgraph of $G_r$ induced by $S$.

Finally, let $S_r$ be the set of vertices occupied by defenders in time step $t$. Formally, $S_r = \{\delta^D_t(d) : d \in D\}$.

APPC is stated as a decision problem as follows:

**Definition 1.** The decision APPC problem: Given an instance $\Sigma = (G, A, D, \alpha_0, \delta^A_0, r)$ of APPC, is there a strategy of destination allocations $\delta^D_t : D \to V$ such that $G_r[S_t]$ is connected for each $t = 0, 1, 2, \ldots$ and such that the team $D$ of defenders is able to prevent all attackers from reaching their targets by moving defending agents according to $\delta^D_t$.

Typically it is not possible to protect all targets. We are therefore also interested in the optimization variant:

**Definition 2.** The optimization APPC problem: Given an instance $\Sigma = (G, A, D, \alpha_0, \delta^A_0, r)$ of APPC, the task is to find a strategy of destination allocations $\delta^D_t : D \to V$ such that $G_r[S_t]$ is connected for each $t = 0, 1, 2, \ldots$ and such that the team $D$ of defenders minimizes the number of attackers that reach their target by moving defending agents according to $\delta^D_t$.

**Destination Allocation**

Since solving APPC in practice is a challenging problem due to its high computational complexity, designed methods are inexact and heuristic. Owing to the large search space we do not consider game-theoretic approaches even though the problem can be regarded as a two-player game. Our solving approaches are based on a technique called destination allocation. The basic idea is to assign a destination vertex to each defender and subsequently use some CPF algorithm adapted for the environment with adversaries, and to lead each defender to its destination while satisfying the connectivity constraint. A defender may be allocated to any vertex, including the attackers' targets. Solving approaches can be divided into two basic categories: single-stage and multi-stage. In single stage methods, targets are assigned to defenders only once at the beginning, as opposed to multi-stage methods, where the destinations can be reassigned any time during the agents' course. In our implementation, once all defenders are allocated to some destinations, they try to get to the desired locations using adapted LRA* algorithm. This work focuses merely on the single-stage methods. In all the studied strategies, every agent is allocated to exactly one location and every location is assigned to at most one defender. Attackers react on defenders by replanning within LRA* if they find that their original path is blocked. Let us recall several destinations allocation strategies and discuss their properties (Ivanova, Surynek, and Hirayama 2018).

**Random Allocation**

For the sake of comparison, we consider a strategy, where each defender is allocated to a random target of an attacker. Neither the agent location nor the underlying grid graph structure is exploited.

**Greedy Allocation**

A greedy strategy is slightly improved approach. It takes the defenders one by one in a random order and allocates them to their closest target of an attacker. The greedy as well as the random strategy do not consider initial locations of attackers and do not exploit the structure of underlying graph in any way. These two methods always allocate defenders to given targets of attackers. The advantage of this approach is that if a defender manages to reach its assigned target, it will never be captured by the attacker aiming for that target. This can be useful in scenarios where the number of defenders is similar to the number of attackers. Unfortunately, such a strategy would not be very successful in instances where attackers significantly outnumber defenders.

**Bottleneck Simulation Allocation**

The idea behind the bottleneck simulation strategy is to gain some information from the map structure and the positions of attackers and assign defenders to vertices that would divert attackers from the protected area as much as possible. The aim is to successfully defend the targets even with a small number of defenders, as illustrated in Fig. 1.

We attempt to identify strategic bottlenecks and block them by defenders. In order to discover bottlenecks of general shape, we develop the following simulation strategy exploiting the underlying grid graph. The basic idea is that as attackers move towards the targets, they are expected to pass through vertices close to a bottleneck more often than through other vertices. This observation suggests to simulate the movement of the attackers and find frequently visited vertices. As defenders do not share the knowledge about paths being followed by attackers, frequently visited vertices are determined by a simulation in which paths of attackers are estimated.

After obtaining such a frequently visited vertex, we then explore its vicinity up to a given distance. If we find out
that there is indeed a bottleneck, its vertices are assigned to some defenders as their new destinations. Under the assumption that the bottleneck is blocked by defenders, the paths of attackers may substantially change. For that reason we estimate the paths again and find the next frequent vertex of which vicinity is searched for bottlenecks. The whole process is repeated until all available defenders are allocated to a destination, or until no more bottlenecks are found. Alg. 1 describes this procedure more formally.

**Algorithm 1: Bottleneck simulation procedure**

```plaintext
Data: G = (V, E), D, A
Result: Destination allocation δ^D
δ^Davailable = D;  // Defenders to be allocated
F = ∅;             // Set of forbidden locations
δ^A = Random guess of δ^A;

while δ^Davailable ≠ ∅ do
  for a ∈ A do
    /* find the shortest path in G between
       an attacker a and its estimated
       target, that avoids passing through
       the forbidden locations in F */
    pa = shortestPath(α^0(a), δ^A(a), G, F);
    end
  f(v) = {pa : a ∈ A ∧ v ∈ pa};  // Frequency of v
  w ∈ arg max_v f(v);
  B = exploreVicinity(w);  // Search a bottleneck
  if B ≠ ∅ then
    D' ∈ δ^Davailable, |D'| = |B|;
    assignToDefenders(B, D');
    δ^Davailable = δ^Davailable \ D';
    F = F ∪ B
  else /* If no new bottleneck is found, assume all have been already discovered */
    break;
  end
  /* If there are some defenders without a
     destination left, they will be allocated randomly */
  assignToRandomTargets(δ^Davailable);

Another approach for bottleneck detection could be based on finding cut-set in a graph. However, our bottleneck simulation prefers bottlenecks frequently used by attackers.

**Connectivity Maintenance**

The requirement of preserving the possibility of communication is modeled by a connectivity maintenance of subgraph of the visibility graph induced by the defenders’ locations. The first task is therefore to create the visibility graph, which depends on the positions of obstacles in the map and a predetermined visibility range. The agents move using an adaptation of the LRA* algorithm that keeps the visibility subgraph connected. Paths are planned such that the first step of a defender must lead to a position which induces a connected visibility subgraph. Defenders follow paths computed by LRA* and whenever an agent is about to enter an occupied location, or if the next move would disconnect the communication subgraph, its path is recalculated.

The movement determined by such an approach will surely maintain the connectivity, however, in many instances, some defenders will not be able to reach the destination locations assigned to them. Our effort is to modify the allocation strategies so that the number of defenders that are not able to reach their assigned destinations is minimized.

Intuitively, defenders should be allocated to their destinations so that in the most optimistic case, when they all reach their desired locations, the connectivity is preserved. This constraint is not guaranteed to be satisfied in general. We propose the following approach to tackle this issue.

Initially, the defenders are partitioned into two sub-sets, communicators Dc and occupiers Do, with a selected ratio |Dc| : |Do|. The occupiers are allocated to attackers’ targets according to one of the allocation strategies described above. In the best scenario, all defenders manage to reach their destinations. It is easy to check, for example by using BFS or DFS on the induced subgraph, whether the ideal final position of defenders maintains connectivity. If the connectivity is violated, the defenders reserved as communicators are allocated to targets so that the subgraph of the visibility graph induced by the defenders’ target locations has as few connected components as possible. Fig. 2 depicts a situation where the three occupiers reached the attackers’ targets assigned to them, but they alone are not able to communicate. Nevertheless, a suitable placement of two communicators ensures that the communication can take place. In fact, the question whether it is possible to allocate destinations for Dc so that the desired position allows a communication among all defenders is already difficult.

**Proposition 1.** Let Σ be an APPC instance with the set of defenders D = Dc ∪ Do, and let δ^D^(d) for each occupier d ∈ Do be already determined. The decision problem whether there exists a destination allocation δ^D = δ^D^(d) for each defender d ∈ Dc such that all defenders maintain connectivity of the visibility graph at their final positions is NP-complete.

**Sketch of proof.** The problem is obviously NP, because checking a connectivity can be done in polynomial time. In or-
order to prove the NP-hardness, we reduce the known NP-complete problem of Vertex Cover (VC) to our problem. Let $H = (V_H, E_H)$ be an instance of VC. For each $e \in E_H$ we create $v_{e} \in V$ such that $v_{e}$ is a destination assigned to some occupier $d \in D_e$. For each $u \in V_H$ we construct $v_{u} \in V$ such that for all $e \in E_H$ incident with $u$ we create $\{ v_{u}, v_{e} \} \in E$. Vertices $v_{u}$ s.t. $u \in V_H$ form a complete subgraph of $G$. Finally, we set $|D_e| = k$. Now $H$ has a vertex cover of size at most $k$ if and only if it is possible to assign destinations to communicators so that the connectivity of $G$ is maintained in the desired position given by $\mathcal{D}_k$.

Let $T_o$ and $T_e$ be the set of targets allocated to occupiers and communicators, respectively. If the induced subgraph $G_e[T_o]$ has several connected components, the used modification of LRA* algorithm could not lead all of them to their targets, because it would cause a loss of communication ability. At this point the set of communicators comes into play. The aim is to find target locations for communicators so that the graph $G_e[T_o \cup T_e]$ is connected. First, the connected components of $G_e[T_o]$ are identified. We then iterate while there are available communicators and connected components to be covered by them. In every iteration, a location $l$ from which a communicator can cover a set of connected components that contains maximum number of targets allocated to occupiers is selected together with the set of covered connected components. The location $l$ is subsequently assigned to the closest unallocated communicator. For a more formal explanation see Alg. 2.

**Algorithm 2: Destination allocation to communicators**

Data: $G_e = (V_e, E_e), D_o, D_e, T_o$

Result: Destination allocation $\mathcal{D}_k$

$T_e = \emptyset$;  // Destinations assigned to communicators

while $D_e \neq \emptyset$ do

$C = \text{connected components of } G_e[T_o \cup T_e]$;

while $C \neq \emptyset$ do

/* A pair of a location $l$ and a subset $C'$ of connected components covered by

$l$ that minimizes the number of vertices in $C'$ */

$(l, C') = \text{arg max}_{(l, C') \in \Phi} \{ \sum_{C \in C'} |C| : \exists v \in C : (v, l) \in E_e \}$;

/* An available agent closest to $l$ */

$a = \text{arg min}_{a \in D_e} \{ |pa(a, l)| \}$;

$\mathcal{D}_k(a) = l$;  // assign destination to agent

$T_e = T_e \cup \{ l \}$;

$C = C \setminus C'$;

$D_e = D_e \setminus \{ a \}$;

if $D_e = \emptyset$ then break;

end

end

**Preliminary Experiments**

The aim of experimental evaluation is to compare individual strategies described above with their counterparts adapted to connectivity maintenance. We would like to find out whether the adaptation improves the success rate of a strategy and also how instance types affect its performance.

Our hypothesis is that when there is a sufficient number of defenders, the adaptation has little or no effect. We predict that in instances, where defenders are outnumbered by attackers, the adaptation increases the success rate of the corresponding strategy. Furthermore, it is likely that the simulation strategy is worse when the connectivity maintenance is required, because the identified bottlenecks may be far from each other, which makes it difficult to preserve communication among them.

We implemented all suggested strategies in Java as an experimental prototype. In our testing scenarios we use maps of different structure with various initial configurations of attackers and defenders. Our choice of testing scenarios is focused on comparing performance of the strategies and discovering what factors have impact on their success.

Different strategies are successful in different types of instances. It is therefore important to design the instances with a sufficient diversity, in order to capture strengths and weaknesses of individual strategies.

**Instance generation and types**

The instances used in the practical experiments are generated using a pseudo random generator, but in a controlled manner. An instance is defined by its map, the ratio $|A| : |D|$ and locations of individual defenders, attackers and their targets. These three entries form an input of the instance generation procedure. Further, we select rectangular areas inside which agents of both teams and the attackers’ targets are placed randomly. The experiments are conducted on 3 different maps that vary in their structure (see Fig. 3).

![Orthogonal rooms](image1)

(a) Orthogonal rooms

![Ruins](image2)

(b) Ruins

![Waterfront](image3)

(c) Waterfront

**Figure 3:** Three different maps used in the evaluation

Each map is populated with agents of 3 different $|D| : |A|$ ratios, namely $1:1$, $1:2$ and $1:5$, with fixed number of attackers $|A| = 50$. The maximum number of moves of the agents is set to 150 for each team. Note that the individual instances are never completely fair to both teams. It is therefore impossible to make a conclusion about a success rate of a strategy by comparing its performance on different maps. The comparison should always be made by inspecting the performance in one type of instance, where we can see the relative strength of the studied algorithms.
Table 1: Average number of agents that eventually reached their target in the map Orthogonal rooms

| | | | | | | |
|---|---|---|---|---|---|
| | RND | RND-C | GRD | GRD-C | SIM | SIM-C |
| 1:1 | 26.0 | 29.0 | 25.5 | 29.1 | 20.8 | 28.3 |
| 1:2 | 41.0 | 39.6 | 39.4 | 40.5 | 29.3 | 31.7 |
| 1:5 | 48.1 | 45.7 | 46.1 | 46.8 | 46.9 | 46.8 |

Experimental results

The following set of experiments compares random, greedy, simulation strategy and their communication counterparts in different instance settings. Each of the following tables contains results associated with one map.

Each entry in the tables shows an average number of attackers that reached their targets at the end of the time limit. The average value is calculated for 10 runs in each settings, always with a different random seed. Random and greedy strategies have very similar results in all positions and team ratios. It is apparent and not surprising that with decreasing |D| : |A| ratio, the strength of defensive strategies decreases.

We focused on evaluation of the effect of using communicating agents in implemented target allocation strategies. For each target allocation strategy we compare the standard version and the version with communicating agents.

Table 1 shows results for Orthogonal rooms map. It can be observed that using communicators is beneficial in case of random strategy where defenders tend to be outnumbered by attackers. On the other hand, communicators cause no improvement in Ruins map (Tab. 2). Small improvement of the bottleneck simulation strategy can be observed in Waterfront map (Tab. 3) again in cases when defenders are outnumbered. Both types of maps where communicators turned out to be beneficial appear to have the structure of large open spaces separated by narrow bottlenecks.

Table 2: Average number of agents that eventually reached their target in the map Ruins.

| | | | | | | |
|---|---|---|---|---|---|
| | RND | RND-C | GRD | GRD-C | SIM | SIM-C |
| 1:1 | 21.5 | 21.1 | 24.8 | 24.7 | 18.3 | 18.6 |
| 1:2 | 42.1 | 40.2 | 39.0 | 40.3 | 37.1 | 36.9 |
| 1:5 | 47.1 | 47.1 | 46.0 | 46.2 | 44.3 | 43.8 |

Table 3: Average number of agents that eventually reached their target in the map Waterfront

| | | | | | | |
|---|---|---|---|---|---|
| | RND | RND-C | GRD | GRD-C | SIM | SIM-C |
| 1:1 | 20.7 | 21.6 | 18.9 | 18.5 | 20.8 | 21.9 |
| 1:2 | 35.2 | 31.2 | 30.7 | 31.4 | 35.8 | 33.5 |
| 1:5 | 41.6 | 41.4 | 40.7 | 40.7 | 42.3 | 41.3 |

Conclusion and Future Work

We have designed several practical algorithms for APPC. We extended previous algorithms for APP with a technique of connectivity maintenance. This is done by dividing defending agents into two groups - occupiers and communicators. The role of occupiers is to protect the area while communicators are placed so that they cover as largest part of the protected area as possible in order to support connectivity among occupiers. Performed experimental evaluation indicates that the effect of using dedicated agents as communicators is much smaller than expected but there is some in maps having the structure of large open spaces separated by bottlenecks.

References

Agmon, N.; Kaminka, G. A.; and Kraus, S. 2011. Multi-robot adversarial patrolling: Facing a full-knowledge opponent. J. Artif. Intell. Res. 42:887–916.

Elmaliach, Y.; Agmon, N.; and Kaminka, G. A. 2009. Multi-robot area patrol under frequency constraints. Ann. Math. Artif. Intell. 57(3-4):293–320.

Haynes, T., and Sen, S. 1995. Evolving behavioral strategies in predators and prey. In Proc. of Adaption and Learning in Multi-Agent Systems, IJCAI’95 Workshop, 113–126.

Ivanová, M., and Surynek, P. 2014. Adversarial cooperative path-finding: Complexity and algorithms. In 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, 75–82.

Ivanova, M.; Surynek, P.; and Hirayama, K. 2018. Area protection in adversarial path-finding scenarios with multiple mobile agents on graphs - a theoretical and experimental study of strategies for defense coordination. In Proceedings of the 10th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, 184–191. INSTICC.

Rubenstein, M.; Ahler, C.; Hoff, N.; Cabrera, A.; and Nagpal, R. 2014. Kilobot: A low cost robot with scalable operations designed for collective behaviors. Robotics and Autonomous Systems 62(7):966–975.

Ryan, M. R. K. 2008. Exploiting subgraph structure in multi-robot path planning. J. Artif. Intell. Res. 31:497–542.

Silver, D. 2005. Cooperative pathfinding. In Proc. of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, 2005, 117–122.

Vidal, R.; Shakernia, O.; Kim, H. J.; Shim, D. H.; and Sastry, S. 2002. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. IEEE Trans. Robotics and Autom. 18(5):662–669.

Wang, K. C., and Botea, A. 2011. MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. J. Artif. Intell. Res. 42:55–90.

Wieselthier, J. E.; Nguyen, G. D.; and Ephremides, A. 2002. Energy-efficient broadcast and multicast trees in wireless networks. Mob. Netw. Appl. 7(6):481–492.