To appear in *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence* (AAAI 11), San Francisco, USA, August 2011.

Multiagent Patrol Generalized to Complex Environmental Conditions

Noa Agmon, Daniel Urieli and Peter Stone *
Department of Computer Science
The University of Texas at Austin
{agmon,urieli,pstone}@cs.utexas.edu

Abstract

The problem of multiagent patrol has gained considerable attention during the past decade, with the immediate applicability of the problem being one of its main sources of interest. In this paper we concentrate on frequency-based patrol, in which the agents' goal is to optimize a frequency criterion, namely, minimizing the time between visits to a set of interest points. We consider multiagent patrol in environments with complex environmental conditions that affect the cost of traveling from one point to another. For example, in marine environments, the travel time of ships depends on parameters such as wind, water currents, and waves. We demonstrate that in such environments there is a need to consider a new multiagent patrol strategy which divides the given area into parts in which more than one agent is active, for improving frequency. We show that in general graphs this problem is intractable, therefore we focus on simplified (yet realistic) cyclic graphs with possible inner edges. Although the problem remains generally intractable in such graphs, we provide a heuristic algorithm that is shown to significantly improve point-visit frequency compared to other patrol strategies. For evaluation of our work we used a custom developed ship simulator that realistically models ship movement constraints such as engine force and drag and reaction of the ship to environmental changes.

1 Introduction

The problem of multiagent patrol has gained considerable attention during the past decade (e.g. (Chevaleyre 2004; Machado et al. 2003; Almeida et al. 2004; Elmaliach, Agmon, and Kaminka 2009; Basilico, Gatti, and Amigoni 2009; Agmon, Kraus, and Kaminka 2008)), with the immediate applicability of the problem being one of its main sources of interest. The problem is formally described as repeatedly visiting some interest points in order to monitor them. The points may either be in a discrete environment, a continuous 1-dimensional environment (along a line), or a continuous 2-dimensional environment (inside an area).[1]

In this paper we concentrate on the continuous 2-dimensional multiagent patrol problem, with discrete points of interest, in complex environmental conditions. In this problem, we are given a graph $G = (V, E)$, and we need to define patrol paths for a team of k agents that will minimize the maximal time some vertex of the graph is left unvisited. The complexity of the environment is expressed via the cost of travel between each pair of vertices of the graph.

Current strategies for multiagent patrol offer, roughly, two alternatives for agents' patrol paths. The first strategy, denoted here as SingleCycle, is to create one simple cyclic path that travels through the entire area (graph), and to let all agents patrol along this cyclic path while maintaining uniform distance between them (Elmaliach, Agmon, and Kaminka 2009; Chevaleyre 2004). The second strategy, denoted by UniPartition, is to partition the area (graph) into k distinct subareas, where each agent patrols inside one area.

We suggest a third, general, strategy, denoted by MultiPartition, in which the graph is divided into m subgraphs, $m \leq k$, such that a subteam of agents jointly patrols in each subgraph. We define the problem of finding k (possibly overlapping) paths for the agents such that the maximal time between any two visits at a vertex is minimized, and show that the problem is $\mathcal{NP}$-Hard. The SingleCycle and UniPartition strategies are special cases of MultiPartition, and are also intractable in general graphs.

We therefore investigate the problem on a special family of graphs, which are cyclic graphs with non intersecting shortcuts (diagonals), called *outerplanar graphs* (Chartrand and Harary 1967). This simplified, yet realistic, family of graphs have some characteristics that can help in finding solutions to the multiagent patrol problem. For example, an optimal SingleCycle strategy is unique and can be found in linear time. Unfortunately, the time complexity of the general problem of finding an optimal MultiPartition strategy even in such graphs appears to be intractable as well. We therefore suggest a heuristic algorithm HeuristicDivide for finding a partition of the graph into disjoint cycles in the outerplanar marine environment, and a partition of the k agents among those cycles.

For evaluation of our work we used a custom developed ship simulator, UTSeaSim, that was designed to realistically model ship movement constraints in marine environments. We first show that in a simple scenario in which the optimal MultiPartition strategy is easily computable, it outperforms the other two strategies (SingleCycle and UniPartition). We then show that in a more complex environment, our heuristic algorithm HeuristicDivide, following the MultiPartition strategy, performs significantly better than the tractable SingleCycle strategy.

2 Related Work

The problem of multiagent patrol can be roughly divided into two problems: multiagent frequency-based patrol (e.g.

*This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

[1]Of course higher dimensions are also possible.

(Machado et al. 2003; Chevaleyre 2004; Elmaliach, Agmon, and Kaminka 2009)), and multiagent patrol in adversarial environments (e.g. (Agmon, Kraus, and Kaminka 2008; Basilico, Gatti, and Amigoni 2009)). The problems differ in the objective function that should be optimized, namely optimizing frequency-based criteria or optimizing probability of detecting events controlled by an adversary (respectively). In this paper we focus on the problem of frequency-based patrol, in which we aim at minimizing the time between two visits at a point.

Mechado *et al.* (2003) were the first to define the problem of multiagent patrol in graph environments, and introduced the notion of *idleness*, meaning the time between two visits in a vertex of the graph. They consider environments with uniform length edges, and perform an empirical evaluation of various architectures for multiagent patrol in different graphs. They did not theoretically define nor evaluate the multiagent patrol problem on graphs.

The first *theoretical* analysis of the problem of multiagent patrol was given by Chevaleyre (2004). Chevaleyre refers mainly to the *worst idleness* criterion, which is the largest amount of time that some vertex remained unvisited throughout the execution of the patrol algorithm. He discusses two possible strategies: a *Cyclic* strategy, in which one cyclic path travels through the entire graph, and all agents follow this path (denoted by SingleCycle) and a *Partition-based* strategy, in which the graph is partitioned into k distinct subgraphs (k being the number of agents), where each agent visits one subgraph in a cyclic tour (denoted by UniPartition). In our work we redefine the multiagent patrol problem in a more general form, in which the graph is possibly partitioned into disjoint subgraphs, however agents can share a subgraph (denoted by MultiPartition). This general definition includes also the two strategies proposed by Chevaleyre as subcases, i.e., SingleCycle, UniPartition $\subseteq$ MultiPartition.

Ahmadi and Stone (2006) investigated the multiagent patrol problem in prioritized environments, i.e., where different areas require different attention from the agents. They suggest a learning-based method for determining the optimal patrol path for each robot, which is adapted to the different constraints of the environment. Marier *et al.* (2010) also consider the prioritized multi-robot patrol, in which they assign non-uniform weights on the vertices of the graph, corresponding to the importance of the node. They describe the problem as information gain (rather than idleness), and examine the performance of their heuristic algorithms with respect to the known (or unknown) duration of the patrol. They do not, however, refer to the graph theoretic problem.

Multi-robot patrol in areas was considered by Elmaliah *et al.* (2009), which offered an optimal patrol algorithm based on a coordinated movement of the robots along one cyclic path with minimal cost that passes through the entire areaTheir solution assumes that the area and the size of the robots meet several constraints, allowing them to find an optimal solution (minimal cost cyclic path) in polynomial time.

3 Motivation - Ship Patrol and Marine Environment

As surveyed in Section 2, the problem of multiagent patrol has become a canonical problem in multiagent (and specifically multi-robot) systems in the past several years. In this paper, we investigate this problem in a realistic ship simulator that we have designed in our lab and that introduces important new travel-time constraints to the problem. The general problem defined in graph environments requires a team of k agents to repeatedly visit all N nodes of the given graph while minimizing the longest time a node has remained unvisited by some robot. Generally, the solutions that exist in the literature for defining optimal patrol paths for a team of robots, can be roughly divided into two types: SingleCycle and UniPartition strategies, which consider the entire cyclic path, or divide the area into k regions, each covered by one agent (respectively).

When looking at the example described in Figure 1 for three ships, we can see that there exists another strategy: Letting one ship patrol in one cycle (here points p_3, p_4, p_5, p_6), and the other two ships can jointly patrol in one cycle (points $p_1, p_2, p_7, p_8, p_9, p_{10}$). We denote this strategy MultiPartition, i.e., a partition into areas in which more than one agent can patrol in each area. In this example, the worst idleness when the sea conditions were calm (no winds or currents) was 651, 786, and 614 seconds for the SingleCycle, UniPartition and MultiPartition strategies (respectively). When we introduced currents to the system, the advantage of using the MultiPartition strategy became more evident: the worst idleness results were 795, 792 and 613 seconds using the SingleCycle, UniPartition and MultiPartition strategies (respectively).

This example, along with other similar phenomena we have viewed in our simulator, motivated us to redefine the problem of multiagent patrol in a more general form, the MultiPartition strategy, and investigate possible solutions to the problem in circular environments, but with additional shortcuts between the points of interest.

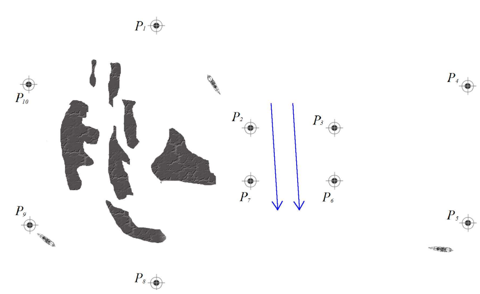

Figure 1: An example of a scenario handled by the simulator. The circles represent the points of interest (nodes of the graph), and the drop shapes are the ships. The large grey shapes are obstacles, and the drawn arrows indicate the direction of the water current.

4 Problem definition and complexity

In this section, we define the general problem of multiagent frequency based patrol on general graphs. We describe the decision version of the problem, where the input is the graph $G = (V, E)$ ($|V| = N$), an integer $k < N$ that corresponds to the number of agents, and an integer f which is the maximal worst idleness, i.e., the maximal requested idleness from all vertices of the graph (similar to the definition in (Chevaleyre 2004)). Formally, if f_i denotes the idleness of a vertex v_i, then the worst idleness of the graph G, $\text{wi}(G)$, guaranteed by an Algorithm $\mathcal{A}$ is defined as $\text{wi}(G) = \max_{1 \leq i \leq N}\{f_i\}$.

Note that real world constraints dictate modeling the world with directed graphs, i.e., the travel time from a vertex v to a vertex u is not necessarily the same as that from u to v. However, we assume that the graph is *symmetric*, i.e., if an edge exists from v to u, then an edge exists also from u to v (not necessarily of the same cost). We therefore describe the general problem on undirected graphs. Once a cycle is defined, the algorithms will decide whether to go clockwise or counterclockwise along the cycle, depending on the direction that has lower cost.

Multiagent patrol in general graphs

Definition: Multiagent Graph Patrol (MGP)
Given a graph $G = (V, E, C)$ where $|V| = N$, and $\forall (v_i, v_j) \in E, c_{ij} \in C$ is the associated cost of the edge, an integer $k < N$ denoting the number of agents, and a desired maximal worst idleness target f, is there a division of V into $m \leq k$ cyclic paths $V_1, V_2, \ldots, V_m$ (not necessarily simple), each assigned with k_i agents such that all k_i agents visit all vertices in V_i and $\sum_{i=1}^{m} k_m = k$, such that the worst idleness $\text{wi}(G)$ is at most f?

In the following theorem we show that the MGP problem is $\mathcal{NP}$ complete for general k's.

Theorem 1. *The* MGP *problem is $\mathcal{NP}$ complete.* [2]

In our work, we would like to consider a special case of the MGP problem, in which each path $V_1, \ldots, V_m$ is cyclic, i.e., it is a closed path with no repeated vertices. Moreover, we restrict our attention to sets of distinct paths that do not share any vertices, i.e, $V = V_1 \bigoplus \ldots \bigoplus V_m$. This problem handles restrictions that are more suitable for realistic robotic environments, in which two requirements are met:

1. Two robots will not meet, thus will not interfere with one another, during the execution of the patrol.
2. Robots will not need to interact outside of their subteam, i.e., the patrol algorithm requires only local coordination (unless the environment changes the optimality of the current patrol algorithm). Moreover, if different human operators observe each subteam, it does not require continuous coordination among the human operators.

The formal definition of the problem is as follows.

Definition: Multiagent Cyclic Graph Patrol (MCGP)
Given a graph $G = (V, E, C)$ where $|V| = N$, and $\forall (v_i, v_j) \in E, c_{ij} \in C$ is the associated cost of the edge, an integer $k < N$ denoting the number of agents and a desired maximal worst idleness target f, is there a division of V into $m \leq k$ *distinct simple* cycles $V = V_1 \bigoplus V_2 \bigoplus \ldots \bigoplus V_m$, each cycle V_i assigned with k_i agents coordinatedly traveling along V_i and $\sum_{i=1}^{m} k_m = k$, such that the worst idleness $\text{wi}(G)$ is at most f?

The MCGP is a special case of the MGP, in which the cyclic paths are required to be disjoint, and each cycle is simple (with no repeated vertices). The $\mathcal{NP}$-Hardness proof resembles the proof for the MGP problem, thus we conclude the following.

[2] Proofs are omitted throughout the paper due to space constraints.

Theorem 2. *The* MCGP *problem on general graphs is $\mathcal{NP}$-Hard.*

We can define the worst idleness in this problem as follows. If k' agents visit a cyclic path, denoted by V^C, where $V^C = \{v_{i_1}, v_{i_2}, \ldots, v_{i_l}\}$, $v_{i_j} \in V(G)$, $(v_{i_j}, v_{i_{(j+1 \mod l)}}) \in E(G)$, and denote the total weight of edges in the cycle by $w(V^C) = \sum_{j=1}^{l} c_{i_j i_{(j+1 \mod l)}}$, then $\forall v_{i_j} \in V^C$, $f_{i_j} = \frac{w(V^C)}{k'}$. Therefore if G is divided into m distinct cycles, where each cycle V_i^C is visited by k_i agents, then $\text{wi}(G) = \max_{1 \leq i \leq m}\{\frac{w(V_i^C)}{k_i}\}$.

Algorithm $AssignKAgents$ (described below) is given as input m cyclic paths, an integer k corresponding to the number of agents, and a maximal idleness f, and has to answer the question of whether k agents are sufficient to guarantee a maximal idleness of f on the given graphs. It returns the assignment of number of agents per graph ($K = \{k_1, \ldots, k_m\}$ such that $\sum_{i=1}^{m} k_i = k$ and k_i agents are necessary to visit G_i in order to guarantee minimal idleness f) and the maximal idleness guaranteed by this assignment (f_{loc}). Denote the edges along the cyclic path G_i in clockwise direction by G_i^{cw} and in the counterclockwise direction by G_i^{ccw}. The algorithm will work for either symmetric directed graphs (in which it will refer to the direction with minimal cost — either going clockwise or counterclockwise) or undirected graphs (in which $w(G_i^{cw}) = w(G_i^{ccw})$ where $w()$ is the cycle weight, or length, function).

Algorithm 1 $< K, f_{loc} >$ = Algorithm AssignKAgents($\{G_1, \ldots, G_m\}, k, f$)

```
C ← 0, K ← ∅
for i ← 1, ..., m do
    w_i ← min{w(G_i^cw), w(G_i^ccw)}
    g_i ← argmin_{G_i^cw, G_i^ccw} {w(G_i^cw), w(G_i^ccw)}
    k_i ← ⌈w_i / f⌉
    if k_i > k then
        Return ∅
    K ← K ⋃ k_i, C ← C ⋃ g_i
    k ← k − k_i
f_loc ← max_{1≤i≤m} {w(C[i]) / K[i]}
Return K, f_loc
```

Multiagent patrol in outerplanar graphs

Motivated by the problem of multi-robot *perimeter patrol* (e.g. (Agmon, Kraus, and Kaminka 2008)), we examine the MCGP problem in circular environments. However, we add more realistic considerations to the environment, namely adding possible *shortcuts* between vertices that pass inside the circle. To avoid possible interference by agents that travel along the edges, we require the inner edges not to intersect one another. The resulting graph is planar, and moreover, it is a *biconnected outerplanar* graph (Chartrand and Harary 1967), i.e., it is a planar graph that is cyclic, and there are no nodes that are inside the cycle (all nodes in the graph are on the same outer face). In the family of outerplanar graphs, several hard problems become very easy to solve. For example finding a Hamiltonian cycle is done in linear time, as the only possible simple cycle that visits all nodes in the graph is the external cycle. Therefore also finding the optimal SingleCycle strategy is done in linear time, as the solution is unique. Another interesting characteristic of outerplanar graphs is that every subgraph of an outerplanar graph is outerplanar, thus a biconnected component of a subgraph of an outerplanar graph also has a unique Hamiltonian cycle (Chartrand and Harary 1967).

We draw a connection between disjoint cycles and biconnected components in Lemma 3. Generally, a biconnected component in an outerplanar graph has a unique Hamiltonian cycle, which is the outer-cycle that visits all vertices. We would therefore like to find a way to use this property in order to find disjoint cycles, as defined in the MultiPartition strategy. As a first step, we look at the case of dividing the graph into two disjoint cycles. We show in the lemma that in order to find such disjoint cycles, it is sufficient and required (in the general case) to consider all biconnected components that are created by the removal of two edges from the graph. We later use this property in the heuristic algorithm for solving the MCGP problem in outerplanar graphs.

Lemma 3. *Given a biconnected outerplanar graph $G = (V, E)$, each division of G into two disjoint biconnected components can be achieved by removing one pair of edges and computing the biconnected components of the remaining graph. If removing one pair of edges, the number of remaining disjoint biconnected components (excluding disconnected vertices) will not exceed* 3.

This lemma results in the fact that finding two disjoint cycles (and possibly 3) in a graph can be done efficiently in time complexity of at most $\binom{|E|}{2}$. Since finding the partition of k into two (or three) components is done efficiently as well, the MCGP problem can be solved optimally in polynomial time if m is restricted to be 2.

Corollary 4. *In an outerplanar graph $G = (V, E)$, finding a division of the graph into up to two disjoint simple cycles V_1^C and V_2^C such that $V = V_1^C \bigoplus V_2^C$ and $wf(G)$ (for any value of k) is minimized can be done in polynomial time, using Algorithm* DivideTo2Cycles.

Algorithm DivideTo2Cycles receives as input the graph $G = (V, E)$ and the maximal frequency criterion f that should be met, and returns the best division of the graphs into two components such that the division results in maximal idleness of at most f. If no such division exists, it returns the empty set. Note that in order to get all possible divisions of G into two disjoint cyclic paths, the algorithm should be given the value $f = w(G)/k$. The algorithm removes all possible pairs of edges from the original graph, and computes the biconnected components of the remaining graph. For those biconnected components, it checks whether there is an assignment of k into those biconnected components such that the maximal idleness of the graph is at most f, using Algorithm AssignKAgents. The optimal assignment of agents can then be done by examining at most $|V|^2$ options of dividing k into two (or three cycles). By Corollary 4, the algorithm examines all possibilities of dividing the graph into two cycles (which has time complexity of $\mathcal{O}(|E|^2)$). Since checking all possibilities of partitioning the number k into at most 3 components is polynomial in k, and determining the idleness is linear in $|E|$, then the total time complexity of Algorithm DivideTo2Cycles is $\mathcal{O}(|E|^3)$.

As shown by de Mier and Noy (2009), the number of cycles in a maximal outerplanar graph is exponential in the number of vertices of the graph, thus if examining all possible sets that generate a direct sum of V it will result in at least an exponential time complexity. We therefore believe (although do not prove) that the MCGP problem, also in the simple biconnected outerplanar environment, is intractable, as the number of all possibilities of the division of the graph into up to k subgraphs grows exponentially with k.

We therefore describe a heuristic algorithm, Algorithm HeuristicDivide, for finding a division of the graph into disjoint cycles.

Algorithm 2 $< U, K, f_{loc} > =$ Algorithm DivideTo2Cycles$(G = (V, E), f, k)$

- $S \leftarrow \emptyset$
- **for** Every pair of edges $e_i = (v_i, u_i)$ and $e_j = (v_j, u_j)$, $e_i, e_j \in E$ **do**
  - $E' \leftarrow E \setminus \{e_i, e_j\}$
  - $U \leftarrow$ biconnected components of $G' = (V, E')$
  - **if** $< K, f_{loc} >=$ AssignKAgents$(U, k, f)! = \emptyset$ **then**
    - $f_{loc} \leftarrow$ optimal assignment of k agents to U
    - $S \leftarrow S \bigcup \{< U, K, f_{loc} >\}$
- Return $S[i]$ for which f_{loc} is minimal

Heuristic algorithm for multiagent patrol in outerplanar graphs

We describe in this section a heuristic algorithm, Algorithm HeuristicDivide, for finding a set of any number of disjoint cycles in an outerplanar graph (based on Algorithm DivideTo2Cycles), and dividing the k agents among these disjoint cycles in a way that aims to find a low overall maximal idleness.

The algorithm applies algorithm DivideTo2Cycles once, then further applies itself recursively on each element of the set of disjoint biconnected components that yield lowest worst idleness. The depth of the recursion is therefore at most k. In this way it performs a greedy heuristic search and halts once it completes all possible divisions up to k cycles. The algorithm receives as input the graph G, the number of agents k, and $f = \frac{w(G)}{k}$.

Note that once the cycles, the direction of travel along the cycles, and the division of the agents among the cycles are determined, it is only left to distribute the agents along the cycles (number of agents per cycle as determined by the algorithm), and require the agents in each cycle to maintain uniform distance between them and continue traveling coordinatedly along their circular path.

Algorithm 3 Algorithm HeuristicDivide$(G = (V, E), f, k)$

- $ResSet \leftarrow$ DivideTo2Cycles(G, f, k)
- **if** $ResSets = \emptyset$ **then**
  - Return G
- $CurChoice = argmin_{< U_i, K_i, f_i > \in ResSet}\{f_i\}$
- Return $\bigcup_{G_i \in U(CurChoice)}$ HeuristicDivide(G_i, K_i, f_i)

Time complexity of Algorithm HeuristicDivide:
The time complexity of HeuristicDivide is defined by two components: The depth of the search and the time to process each level of the search tree. Since at each step we deepen the recursion we lose at least one vertex (as the minimal division to two distinct cycles is to a vertex v and to a cyclic

graph $G \setminus \{v\}$), the depth of the tree is at most $k-1$. The time complexity of AssignKAgents is linear in the number of edges, and the complexity of finding biconnected components is also linear in outerplanar graphs. Therefore the complexity of examining each pair of edges is $\mathcal{O}(|E|)$. When we go down in the recursion, if E is divided into up to three disjoint biconnected components E_1, E_2 and E_3, then we have complexity of $\mathcal{O}(|E_1|^2 + |E_2|^2 + |E_3|^2)$ where $0 \leq |E_1| \leq |E_2| \leq |E_3| < |E|$ and $|E_1| + |E_2| + |E_3| = |E|$. Therefore, since $\mathcal{O}(|E_1|^2 + |E_2|^2 + |E_3|^2) < \mathcal{O}(|E|^2)$ it leads to a total time complexity of $\mathcal{O}(k|E^2|)$. Added to the time complexity of DivideTo2Cycles, the time complexity of HeuristicDivide is $\mathcal{O}(|E|^3)$.

5 Empirical evaluation

We evaluated HeuristicDivide under realistic marine environment using our novel UTSEASIM Simulator. In the following section we describe the simulator, the experimental setting and the empirical results from executing the patrol algorithms described above.

The UTSEASIM Simulator

For our experiments, we use a custom-designed naval surface navigation simulator that will soon be open-source, that we call UTSEASIM. The UTSEASIM simulator is written as a supporting platform for research on autonomous sea navigation. It uses realistic $2D$ physical models of marine environments and sea vessels, and runs both in GUI and in non-GUI modes. Figures 1 and 2 show snapshot of the simulator's GUI (with blue highlights added), taken during a real-time simulation.

The simulator's core contains three main modules: a *Sea Environment* module, a *Ship* module, and a *Decision-Making* module. The sea environment module includes models of winds, water currents, waves, and obstacles. The ship module models all relevant aspects of a ship, including the ship's physical properties, sensing capabilities, and ship actuators. The decision making-module implements an agent that controls a ship autonomously. At each time step, the agent receives the perceptions sensed by the ship, processes them to update its current world state, and decides on control actions for the ship based on its current world state and its decision-making strategy.

Experimental Setting and Results

In our experiments, we examined several different environments. Due to space constraints, we describe one interesting environment, in which the graph is outerplanar with a large number of possible division of the graph into disjoint cycles. The marine environment was implemented in UTSEASIM, and illustrated in Figure 2 (distances are shown in meters). In this environment, $N = |V| = 36$, and the number of ships (k) varies from 1 to 30. Although the points of interest are arranged in two straight lines, this environment can become equivalent to many realistic cases by controlling the currents between nodes thus controlling the edge lengths. Moreover, this environment can represent a man-made group of points of interest, for instance a sequence of oil rigs.

We examined four different scenarios, where in each one the sea conditions were different. In the first environment (*Sea Condition* 0), there were no currents and the cost of traveling between two points correlates to the geographical distance. We then gradually added more currents to the system with three different strengths - *Sea Condition* $1, 2$ *and* 3, for weak, medium and strong, respectively, where their directions was as shown in Figure 2. We ran Algorithm HeuristicDivide initially with the worst idleness of ∞ and let it find the best division it can. We then simulated the patrol-division returned by the algorithm for 20,000 seconds (333 minutes), and reported the worst node-idleness, i.e., the highest average time between consecutive node visits, for any node. In all the results, lower values are *better*.

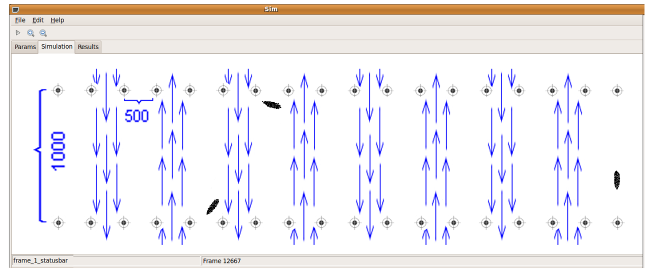

Figure 2: The evaluation environment.

Figure 3 shows the worst idleness resulting from HeuristicDivide when running on sea conditions 0–3. Note that in Sea Condition 0 (no currents) the algorithm always returned the SingleCycle strategy, which is indeed the best division for the case of no currents. As expected, the worst idleness becomes higher as the sea conditions become rougher.

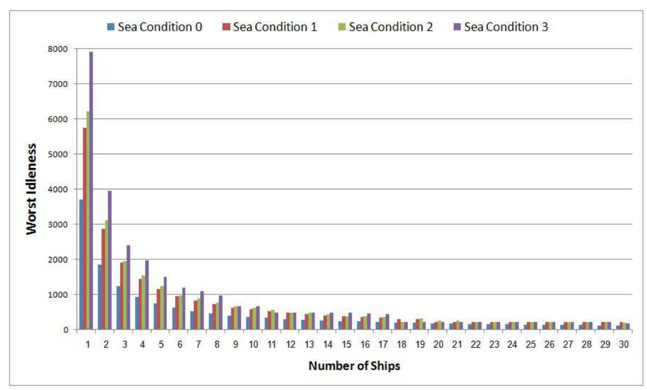

Figure 3: The worst idleness returned from Algorithm HeuristicDivide in the four different sea conditions examined.

In order to evaluate the performance of HeuristicDivide, we compared its worst idleness with the worst idleness of the following:

SingleCycle The patrol algorithm in which all ships travel along one cycle while maintaining uniform distance between adjacent pairs.

k+1 Incremental change. In this case, HeuristicDivide was computed for k ships, and upon the arrival of a new ship it is added to the cycle with highest worst idleness (for the best improvement in idleness). This is compared to running HeuristicDivide with $k+1$ ships, and the goal is to check whether HeuristicDivide, as a heuristic algorithm, does better than just a straightforward increment.

k-1 Decremental change. Here HeuristicDivide was computed for $k+1$ ships, and a ship needs to be removed from the system. We assume that in this case a ship is removed from the cycle with best worst idleness (for minimal increase in the worst idleness). This is compared to running HeuristicDivide with $k-1$ ships, again with the goal of checking whether HeuristicDivide does better than just a straightforward adjustment.

Note that our solution could not be compared against the UniPartition strategy, as the calculation of the optimal strategy is intractable (even for $k = 2$).

The results for Sea Conditions 3 are shown in Figure 4. In Sea Condition 0, HeuristicDivide's solution is always the SingleCycle strategy, hence for every given k HeuristicDivide's solution is identical to the SingleCycle, $k+1$ and $k-1$ solutions. In Sea Conditions 2 and 3, HeuristicDivide performs statistically significantly (using paired t-test) better than the SingleCycle strategy and the incremental $k+1$ and decremental $k-1$ cases, with p-values < 0.002 for Sea Condition 3 (in all cases) and p-value < 0.04 for Sea Condition 2 (in all cases). In Sea Condition 1, although the results are generally better, no significant results are achieved. Interestingly, there are some cases in which SingleCycle slightly outperforms HeuristicDivide, even though HeuristicDivide (theoretically) does not divide a cycle into smaller cycles unless it improves the worst idleness (observed mainly in Sea Condition 1). The reason for that lies in the fact that deciding whether to break a big cycle into smaller cycles is done using an estimation of the cost of traveling along an edge, by averaging across simulations of ships patrolling along the edges of arbitrary paths, which are usually different then the paths found by HeuristicDivide's solution. For instance, consider the case where the final solution requires a $180°$ turn towards the next point. The physics of ship movement require the ship to travel in an arc, which makes the path to the next point longer than its estimate. We leave the incorporation of the cost of traveling along angular paths in HeuristicDivide to future work.

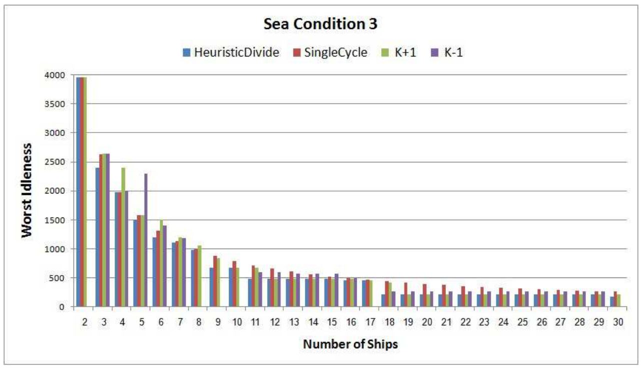

Figure 4: The worst idleness results of HeuristicDivide, SingleCycle, $k+1$ and $k-1$ in Sea Condition 3 (strong currents).

6 Conclusions and Future Work

In this paper we introduced a new class of strategies for the frequency-based multiagent patrol problem suitable for multiagent patrol in complex environmental conditions. In this new strategy class, which we call MultiPartition, a graph is divided into disjoint cycles in which a subteam of the agents travel coordinatedly such that the maximal time between visits to interest points is minimized. This strategy class is a generalization of existing strategies that either create one cyclic path throughout the entire graph (SingleCycle strategies) or divide the graph into k disjoint subgraphs (k being the number of agents), where each agent patrols in its own subgraph—the UniPartition strategy. We showed that finding an optimal strategy to the problem is $\mathcal{NP}$-Hard in the general case, and also intractable in a realistic simplified family of outerplanar graphs. We then introduced a heuristic algorithm that divides the outerplanar graph into disjoint cycles. We evaluated our heuristic algorithm in a custom-developed ship simulator that realistically models ship movement constraints such as engine force and drag, and reaction of the ship to environmental changes. Results indicate that this algorithm significantly improves the frequency of visits compared to other known patrol strategies.

This paper opens up several directions for future work. First, it would be interesting to consider the problem of multiagent patrol in prioritized environments, i.e., where vertices of the graph should be visited with different frequencies. Second, we intend to add more learning methods for determining the cost of travel, especially in prioritized environments. From a larger perspective, this paper raises the challenge of finding effective heuristics, central and local, for the MultiPartition problem on general graphs.

References

Agmon, N.; Kraus, S.; and Kaminka, G. A. 2008. Multi-robot perimeter patrol in adversarial settings. In *Proc. of ICRA'08*.

Ahmadi, M., and Stone, P. 2006. A multi-robot system for continuous area sweeping tasks. In *Proc. of ICRA'06*.

Almeida, A.; Ramalho, G.; Santana, H.; Tedesco, P.; Menezes, T.; Corruble, V.; and Chevaleyr, Y. 2004. Recent advances on multi-agent patrolling. *Lecture Notes in Computer Science* 3171:474–483.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proc. of AAMAS'09*.

Chartrand, G., and Harary, F. 1967. Planar permutation graphs. *Annales de l'institut Henri Poincare' (B) Probabilite's et Statistiques* 3(4):433–438.

Chevaleyre, Y. 2004. Theoretical analysis of the multi-agent patrolling problem. In *Proc. of IAT'04*.

de Mier, A., and Noy, M. 2009. On the maximum number of cycles in outerplanar and series-parallel graphs. *Electronic Notes in Discrete Mathematics* 31:489–493.

Elmaliach, Y.; Agmon, N.; and Kaminka, G. A. 2009. Multi-robot area patrol under frequency constraints. *Annals of Math and Artificial Intelligence journal (AMAI)* 57(3—4):293—320.

Machado, A.; Ramalho, G.; Zucker, J.; and Drogoul, A. 2003. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Proc. of MABS'02*, 155–170.

Marier, J.; Besse, C.; and Chaib-draa, B. 2010. Solving the continuous time multiagent patrol problem. In *Proc. of ICRA'10*.