2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

An Iterative Mixed Integer Linear Programming Approach to Pursuit Evasion Problems in Polygonal Environments

Johan Thunberg and Petter Ögren

Abstract— **In this paper, we address the multi pursuer version of the pursuit evasion problem in polygonal environments. It is well known that this problem is NP-hard, and therefore we seek efficient, but not optimal, solutions by relaxing the problem and applying the tools of Mixed Integer Linear Programming (MILP) and Receding Horizon Control (RHC).**

Approaches using MILP and RHC are known to produce efficient algorithms in other path planning domains, such as obstacle avoidance. Here we show how the MILP formalism can be used in a pursuit evasion setting to capture the motion of the pursuers as well as the partitioning of the pursuit search region into a cleared and a contaminated part. RHC is furthermore a well known way of balancing performance and computation requirements by iteratively solving path planning problems over a *receding* planning horizon, and adapt the length of that horizon to the computational resources available. The proposed approach is implemented in Matlab/Cplex and illustrated by a number of solved examples.

I. Introduction

The visibility based pursuit evasion problem addressed here was first proposed by Suzuki and Yamashita [9] and later studied in e.g., [2]–[5]. The problem is to find a search strategy for a group of pursuers, such that an evader moving arbitrarily fast, and starting in an unknown location, will be captured no matter what path he decides to take.

The obvious applications of the pursuit evasion problem is where security guards, or robots such as the one in Figure 1, are to clear an office, a warehouse, or a shop after closing time. However, search strategies of this type can also be used in search and rescue missions, or when looking for an item that might be moved by a non-adversarial agent in a larger area, such as a warehouse.

A complete solution to the one-pursuer case was proposed by Guibas et al. [3] where it was also pointed out that the extension of that same approach presented considerable challenges even in the two-pursuer case. This is natural, since Guibas et al. also showed that the general problem is indeed NP-hard, a fact that essentially removes the hope of finding optimal solutions in reasonable time. The concepts of [3] were built upon in [2], where a field of view limitation was incorporated into the problem. The one-pursuer case was successfully treated, but once again, the multiple-pursuer case turned out to be computationally intractable.

The first author would like to gratefully acknowledge the financial support by Nationellt Rymdtekniskt Forskningsprogram (NRFP).

J. Thunberg is with the Division of Optimization and Systems Theory, Department of Mathemathics, Royal Institute of Technology (KTH), Stockholm, Sweden SE-100 44, johan.thunberg@math.kth.se

P. Ögren is with the Department of Aeronautical and Systems Technology, Swedish Defence Research Agency (FOI), Stockholm, Sweden, SE-164 94, petter.ogren@foi.se

Fig. 1. The security robot Groundbot, developed by Rotundus (www.rotundus.se), during a demonstration of our earlier results, reported in [8].

As optimal deterministic strategies with guaranteed capture are hard to find, the option of using randomized strategies was explored in [5]. It was shown that a single pursuer can locate an evader in any simply connected environment with high probability. Randomized approaches such as these are clearly an option for the multi-pursuer problem, but will not be investigated here.

A closely related problem is the one where the evader and pursuers are constrained to move in a graph. One version of this problem is called the GRAPH-CLEAR problem, and was studied in [6]. In the GRAPH-CLEAR problem, each vertex corresponds to a room, and each edge corresponds to a door. Each vertex and edge furthermore has a number assigned to it, corresponding to how many pursuers are needed to clear the vertex (room), or block the edge (door). The problem is now to deploy pursuers to the edges and vertices in such a way that the whole graph is cleared. It is easy to see how most polygonal environments can be divided into rooms and doors. Therefore, a hierarchical approach with a global GRAPH-CLEAR problem and a polygonal pursuit evasion problem for each room can be created. The benefits of such an approach is to reduce the size of the subproblems, while the potential drawback is that the possibility of a pursuer to see from one room to another is removed.

In this paper, we will propose an approach using the tools of Mixed Integer Linear Programming (MILP) and Receding Horizon Control (RHC). These tools were applied to UAV path planning in [7], where MILP was used to find detailed trajectories over a short planning horizon. The MILP computations were then iterated in a RHC fashion where each trajectory ended closer to goal than the previous one. The polygonal pursuit evasion problem is quite different from the UAV problem studied in [7], but share the properties of a complex short term planning step and a long term goal. In the pursuit evasion problem we let the size of the cleared area be a measure of how far we are from completing the search, and encode the motion of the pursuers and the evolution of the cleared area into a MILP problem that is iteratively solved. The main contribution of the paper is this MILP formulation of the visibility based pursuit evasion game. To the best of our knowledge, this has not been done before.

The outline of the paper is as follows. In Section II the polygonal pursuit evasion problem is formally stated and in Section III the proposed solution is described. Then, Section IV describes an RHC extension of the algorithm. Finally, Section V contains simulation examples to illustrate the approach and Section VI concludes the paper.

II. Problem formulation

Following Guibas et al. [3], the pursuers and evader are modeled as points moving in the polygonal free space, F. Let $e(\tau)$ denote the position of the evader at time $\tau \geq 0$. It is assumed that $e : [0, \infty) \rightarrow F$ is continuous, and the evader is able to move arbitrarily fast. The initial position $e(0)$ and path e is not known to the pursuers. At each time instant, F is partitioned into two subsets, the cleared and the contaminated, where the latter might contain the evader and the former might not. Given N pursuers, let $p_i(\tau) : [0, \infty) \rightarrow F$ denote the position of the i:th pursuer, and $P = \{p_1, \dots, p_N\}$ be the *motion strategy* of the whole group of pursuers.

Let $V(q)$ denote the set of all points that are visible from $q \subset F$, i.e., the line segment joining q and any point in $V(q)$ is contained in F.

Problem 1 (Pursuit Evasion): Given an evader, a set of N pursuers and a polygonal free space F, find a solution strategy P such that for every continuos function $e : [0, \infty) \rightarrow F$ there exists a time τ and an i such that $e(\tau) \in V(p_i(\tau))$, i.e., the pursuer will always be seen by some evader, regardless of its path.

It was shown in [3] that computing the minimal number of pursuers needed to solve Problem 1 is NP-hard. Hence it is also NP-hard to determine if a solution exists for a given number of pursuers. To find efficient solutions in reasonable time one must thus sacrifice optimality. This can be done by exploring randomized approaches [5], or by relaxing the problem and applying other optimization schemes.

In the following section we will first relax Problem 1 by discretizing it, and then apply a combination of Mixed Integer Linear Programming (MILP) and Receding Horizon Control (RHC). These tools have proved to be very useful when addressing other hard path planning problems [7] and we will argue that they are applicable to Problem 1 as well.

III. Proposed Solution

In the proposed solution, we first discretize Problem 1 by partitioning the polygonal free space F into a set of *convex* regions, $F = \cup_{i \in J} F_i$, $J = \{1, \dots, K\}$. The relations between those regions are then described by $M_j \subset J$ and $N_j \subset J$, where M_j is the index set of other regions that are neighbors to F_j, and N_j is the index set of regions that are visible from F_j. Then, a MILP is formulated, capturing what regions are occupied by pursuers at what times, and when the regions are cleared or contaminated over time. By maximizing the cleared area at the end time of the MILP, the continuous pursuer trajectories $p_i(\tau)$ can be constructed from the discrete MILP output.

A. Discretization of the free space environment

The first step of the discretization of Problem 1, i.e., the partitioning $F = \cup_i F_i$, is illustrated in Figure 2. As can be seen, all straight obstacle boundaries are extended until they reach another obstacle, or the perimeter of F. These extended boundaries form the partition $F = \cup_i F_i$.

Lemma 1: In the partition there are $O(n^2)$ regions, where n is the number of straight boundaries of the obstacle polygons and the perimeter.

Proof. In the partitioning, each extended straight obstacle boundary intersects a number of other extended boundaries. There are at most n^2 such intersections. Each such intersection borders at most four regions. Thus the number of regions k, satisfy $k \leq 4n^2$ and $k \in O(n^2)$. ∎

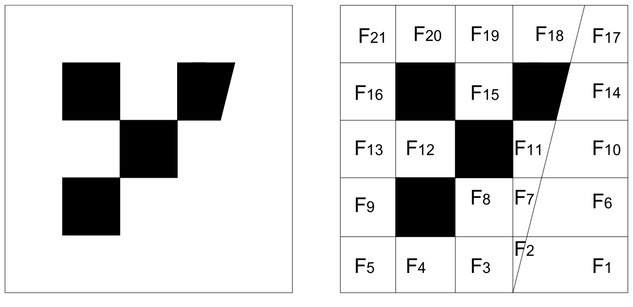

Fig. 2. An example environment with one irregularly shaped obstacle (a), and the corresponding partition of the free space F into convex polygons $F_1 \dots F_{21}$ (b).

The second step of the discretization of Problem 1 deals with the motion, $p_i(\tau)$, of the pursuers. These are now discretized into moving between the regions F_i. A pursuer standing in F_i can in the next, discretized, time instant occupy any region with index in the set M_i, i.e., any neighboring region. This is illustrated in Figure 3 (a).

The third step in discretizing Problem 1 involves the visible set $V(\cdot)$. Let N_i be the index set of regions such that $F_j \subset V(x)$ for all $j \in N_i$ and all $x \in F_i$. Note that visibility is symmetric, i.e. $j \in N_i$ implies $i \in N_j$.

Remark 1: Note that the discretization of $V(\cdot)$ is conservative, since regions F_j that are partially visible are

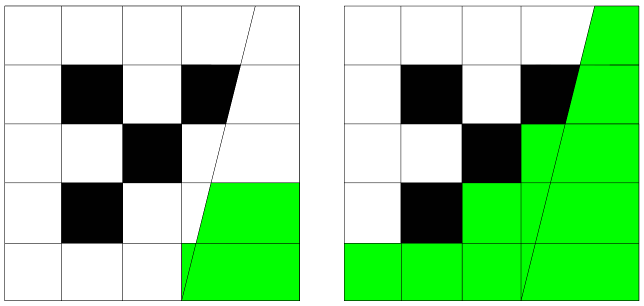

Fig. 3. The neighborhood that can be moved to, M_1 (a) and the neighborhood that can be seen, N_1 (b), from area F_1.

considered not visible at all. In other approaches, such as [2], [3], this is not the case.

The final step of discretizing Problem 1 is to capture the regions being clear, or contaminated during the search in terms of a MILP.

B. MILP formulation

As described above, a pursuer located in polygon i sees the polygons with index in set N_i and can move to polygons with index in the set M_i.

During the search we keep track of where the pursuers are, and which polygons are cleared and which are contaminated. In order to do so, we introduce the following binary variables $\lambda_{it}, \sigma_{it}, \theta_{it} \in \{0,1\}$, where $i \in J$ and $t \in \{1, 2, ..., T\}$. Let $\lambda_{it} = 1$ if and only if a pursuer is *located* in polygon i at time t. Let furthermore $\sigma_{it} = 1$ if and only if polygon i is *seen* at time t and $\theta_{it} = 1$ if and only if polygon i is *cleared but unseen* at time t.

Before formulating the MILP we define four different search-states that each region F_i can be in. Theoretically, there are eight combinations of the three binary variables $\lambda_{it}, \sigma_{it}, \theta_{it}$, but given the meanings we assign to them, only four of those eight combinations are possible, and we denote them S_1, S_2, S_3, S_4. These four states will help us capture the time evolution of the search in the MILP formalism. We differentiate between three different *cleared* states, S_1, S_2, S_3 and one *contaminated* state, S_4.

S_1 The region is seen by a pursuer and contains a pursuer, i.e., $\lambda_{it} = 1,\ \sigma_{it} = 1$ and $\theta_{it} = 0$.

S_2 The region is seen by a pursuer, but does not contain a pursuer, i.e., $\lambda_{it} = 0,\ \sigma_{it} = 1$ and $\theta_{it} = 0$.

S_3 The region is not seen by a pursuer, but can not contain the evader, i.e., $\lambda_{it} = 0,\ \sigma_{it} = 0$ and $\theta_{it} = 1$.

S_4 The region might contain the evader, i.e., $\lambda_{it} = 0,\ \sigma_{it} = 0$ and $\theta_{it} = 0$.

Note that no other combinations of $\lambda_{it}, \sigma_{it}, \theta_{it}$ are possible by definition.

We now state the MILP formulation and then show, in Lemma 2, that a feasible solution does indeed correspond to traversable pursuer paths $p_i(\tau)$ and an expanding cleared region $\{i : \theta_{it} = 1\}$. Note that the proof of Lemma 2, as a side effect, gives motivations for all the constraints (2)-(12).

Problem 2 (MILP): Given a $T \in \mathbb{Z}^+$ solve the following integer linear program.

$$\max\ Z = \alpha \sum_{i \in J} \theta_{iT} + (1-\alpha) \sum_{i \in J} \sigma_{iT} \tag{1}$$

subject to

$$\begin{aligned}
\sum_{j \in N_i} \lambda_{jt} - \sigma_{it} &\geq 0, & (2)\\
\sigma_{it} - \lambda_{jt} &\geq 0 \quad \forall j \in N_i & (3)\\
\sum_{i \in J} \lambda_{it} - N &= 0 & (4)\\
\sum_{j \in M_i} \lambda_{jt} - \lambda_{i(t-1)} &\geq 0 & (5)\\
N - (N-1)\lambda_{it} - \sum_{j \in M_i} \lambda_{jt} &\geq 0 & (6)\\
2 - \sum_{j \in M_i} \lambda_{j(t-1)} - \lambda_{it} &\geq 0 & (7)\\
2 - \sum_{j \in M_i} \lambda_{jt} &\geq 0 & (8)\\
\sigma_{jt} + \theta_{jt} - \theta_{it} &\geq 0, \quad \forall j \in M_i - \{i\}, & (9)\\
\sigma_{i(t-1)} + \theta_{i(t-1)} - \theta_{it} &\geq 0, & (10)\\
1 - \sigma_{it} - \theta_{it} &\geq 0, & (11)\\
\theta_{i1} &= 0, & (12)
\end{aligned}$$

where $\alpha \in [0, 1]$, $i \in J$ and $t \in \{2, 3, ..., T\}$ in (5), (7) and (10) and $t \in \{1, 2, ..., T\}$ in the other constraints.

Note that $\alpha = 1$ corresponds to maximizing the cleared but unseen region (S_3), $\alpha = 0$ corresponds to maximizing the visible region (S_1 or S_2), while $\alpha = 0.5$ corresponds to maximizing the cleared region (S_1, S_2 or S_3) at the final time T. In Section V below we will see that $\alpha = 1$ is actually the best measure of progress for the clearing task. Note also that constraint (4) implicitly assumes that pursuers never occupy the same region. This restriction is somewhat conservative, as it is not present in Problem 1.

Lemma 2: A feasible solution to Problem 2 can be used to generate pursuer paths $p_i(\tau), \tau \in [0, T'], i \in \{1, 2, ..., N\}$, guaranteeing the following. If $e(\tau) \notin V(p_i(\tau))$ for all i and $\tau \in [0, T']$, then $e(T') \in F_i$ such that F_i is in state S4 at time T, i.e., if the evader has not been seen up till time T', then it must be in the contaminated area. Above, T' is the continuous final time corresponding to the discrete final time T.

Proof. We first prove that N valid pursuer paths can be generated from a feasible solution. In (4) it is guaranteed that there are exactly N pursuers at each time t. In (5) it is guaranteed that there must must be a pursuer in the move neighbourhood of polygon i at time $t+1$ if there is a pursuer at the polygon i at time t. Constraints (6), (7) and (8) together guarantee that a pursuer move between adjacent regions in consecutive time steps. Now, pursuer paths $p_i(\tau)$ can be created from λ_{it} where all pursuers cross borders between the F_i at the same time. Finally, a mapping between continuous time τ and discrete time t can be created to accommodate the pursuer velocity bounds.

To see that the right regions are denoted as seen, $\sigma_{it} = 1$, we note that in (2) and (3) the variable σ_{it} is set to 1 if and only if there is a $j \in N_i$ such that $\lambda_{jt} = 1$.

To see that the cleared area, $\theta_{it} = 1$, evolves correctly note the following. In (9) it is verified that the polygon i cannot be in state S_3 at time t if any of the M_i-neighbours are in state S_4, and in (10) it is verified that the polygon i cannot be in state S_3 at time t if it was in state S_4 at time $t-1$. In (11) it is verified that polygon i cannot be in state S_3 if it is in state S_1 or state S_2 at time t and (12) verifies that that no polygon is in state S_3 when the search starts.

To conclude we note that evader $e(\tau)$ can not start in the cleared area S_3 and that the cleared area is always separated from the contaminated S_4 by seen or occupied areas S_1, S_2. Thus, if it is not seen, it must be in the contaminated area. ∎

Lemma 3: A feasible solution strategy P to Problem 2 with N pursuers ending with an empty contaminated area, i.e.,

$$\sum_{i \in J} (\sigma_{iT} + \theta_{iT}) \quad = \quad card(J), \tag{13}$$

is a solution to Problem 1.
Proof. A straightforward application of Lemma 2 above. ∎

Remark 2 (Backwards): Given a solution strategy P of Problem 1, a new solution can be created by running the pursuer trajectories $p_i(\tau)$ backwards. To see this note that the cleared unseen area (S_3) is always separated from the contaminated area (S_4), and we start with an empty cleared unseen area and finish with an empty contaminated area. Running the trajectories backwards would thus result in exchanging the labels cleared unseen and contaminated, i.e. switching states S_3 and S_4.

Remark 3 (Number of pursuers): In the proposed MILP formulation, the number of variables or the number of constraints will not increase with the number of pursuers, *i.e.*, the size of the problem does not grow with the number of pursuers. However, the number of constraints does grow linearly with the number of regions.

IV. Reducing the computation times using RHC and relaxation

In this section we will describe how the computation times for solving Problem 1 can be reduced using RHC and relaxing some of the integer constraints in the MILP.

A. An RHC Solution to the Pursuit Evasion Problem

Depending on the problem size, large time horizons T might be needed to find a solution to Problem 2 with empty contaminated area, and large time horizons T often result in long computation times. As explained in Section I above, a classical way to balance performance with computational resources is RHC, where an optimization problem over a shorter time horizon is iteratively solved instead of solving it once over a longer horizon. In our setting the RHC concept might be implemented as follows.

Algorithm 1:

1) Solve the MILP with $\alpha = 1$ or $\alpha = 0.5$ and some given horizon length T.
2) If the final states σ_{iT} and θ_{iT} satisfies
$$\sum_{i \in J} (\sigma_{iT} + \theta_{iT}) \quad = \quad card(J), \tag{14}$$
the whole area is cleared, and the algorithm terminates.
3) Else, if there was no increase in $\sum_{i \in J}(\sigma_{iT} + \theta_{iT})$ increase either the horizon length T or the number of pursuers N.
4) Prepare a new RHC iteration by removing constraint (12) and adding constraints setting the initial states of the next iteration $\theta_{i1}, \lambda_{i1}, \sigma_{i1}$ equal to the terminal states $\theta_{iT}, \lambda_{iT}, \sigma_{iT}$ of the current iteration.
5) Goto 1.

Remark 4: In step (3) of Algorithm 1 it is preferable to first increase the time horizon, and if this does not work increase the number of pursuers.

Lemma 4: If Algorithm 1 terminates, a solution to Problem 1 is found.
Proof. A straightforward application of Lemma 3 above. ∎

B. Relaxation of the MILP Problem

To increase the computational efficiency when solving Problem 2 we note that some of the integer constraints can be relaxed.

Problem 3 (Relaxation): The variables σ_{it} and θ_{it} in Problem 2 are relaxed such that they are no longer binary variables but belong to $[0, 1]$, *i.e*

$$0 \leq \quad \sigma_{it} \quad \leq 1, \tag{15}$$
$$0 \leq \quad \theta_{it} \quad \leq 1. \tag{16}$$

Using CPLEX 10.2, Problem 3 seems to be twice as fast as the original formulation.

Lemma 5: The pursuer paths λ_{it} in the solution to Problem 3 are also pursuer paths in an optimal solution to Problem 2.
Proof. Note that if there is a j such that $\lambda_{jt} = 1$, $j \in N_i$ then $\sigma_{it} = 1$ by (3), also if $\lambda_{jt} = 0$, $\forall j \in N_i$ then $\sigma_{it} = 0$ by (2), thus σ_{it} is binary. Now let (λ, σ, θ^b) be a (possibly sub-optimal) solution to to Problem 2 in which only θ^b differs from the solution of problem 3. Let Z_2 and Z_3 be the cost of the solutions to Problem 2 and 3 respectively. By (9), (10) and (11) we get that that for each $\theta_{it} \in (0, 1]$, $\theta^b_{it} = 1$. This implies that $Z_3 \leq Z_2$, but since the space that the variables live in in problem 2 is a subset of the space in problem 3, i.e., the latter is a relaxation, $Z_2 \leq Z_3$. Thus $Z_2 = Z_3$. ∎

V. Simulation Examples

When running Algorithm 1, it turns out that the best results are found using $\alpha = 1$. A drawback of the more intuitive $\alpha = 1/2$ is that the pursuers might get stuck at positions where they see a large area, e.g., looking down a corridor, but any motion results in a reduction of this area. Thus we use $\alpha = 1$ in all but one of the examples below. The simulations were done on a Intel Xeon CPU X5450, 3.00GHz with 4 cores, running the MILP software CPLEX 10.2 [1]. Furthermore, all results were found using the relaxed version, Problem 3, as it was found to be on average twice as fast as the non relaxed formulation.

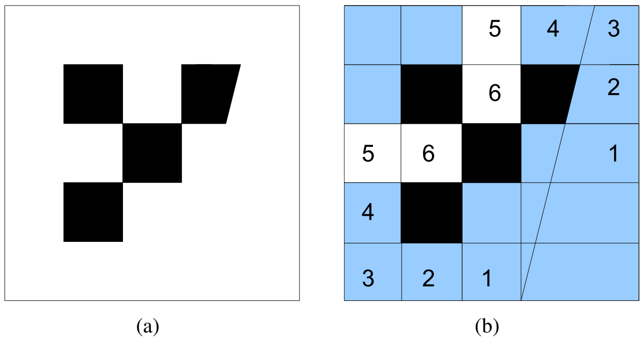

Fig. 4. The results of running Algorithm 1 with a 6 step planning horizon (b) in the environment in (a). The color coding of the sets F_i corresponds to their state at the end of the execution and the numbers denote pursuer positions at corresponding time steps $1 \dots 6$. Blue regions are in state S_3 and white regions are in state S_1 or S_2.

The first problem instance is depicted in Figure 2 with the corresponding solution in Figure 4. With a time horizon of $T = 6$, and $\alpha = 1/2$, a single RHC iteration was needed, and found in 4 seconds.

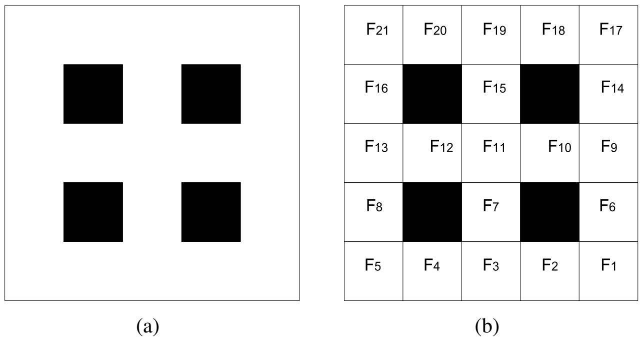

Fig. 5. The partition (b) of a manhattan grid with four obstacles (a).

The partitioning of the second problem instance with four rectangular obstacles is shown in Figure 5. The problem was first solved in 3 RHC iterations using a total number 10 time steps. The computational time was about 3 seconds for each iteration resulting in a computational time of 9 seconds in total. These solutions are depicted in Figure 6 (a-c). Note that the first two RHC iterations achieved progress with $T = 3$, while the third iteration needed $T = 4$ to remove the last S_4 region. Figure 6 (d) shows the result of the iteration leading to the increase in horizon length. This problem was also solved in a single iteration using a time horizon of $T = 6$, with a corresponding computation time of 110 seconds.

The third problem instance is shown in Figure 7 with corresponding solution in Figure 8. The solution involves three RHC iterations with 2 pursuers, followed by one iteration with three pursuers. The computational time was about 5 seconds for the three first iterations and 15 seconds for the last iteration. Trying to find a 2 pursuer solution we

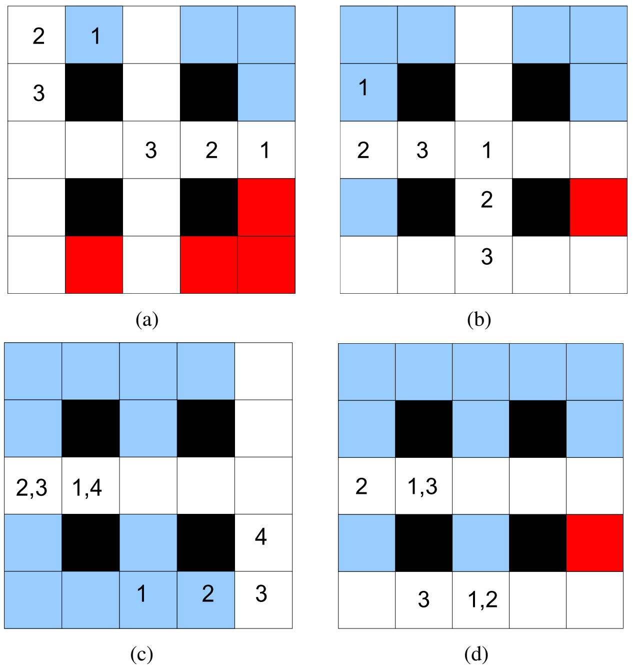

Fig. 6. The results of running the algorithm on a manhattan grid with four obstacles. The partition is showed in Figure 5, and the search problem is solved with 2 pursuers in three iterations where the results of iteration 1, 2 and 3 are shown in (a), (b) and (c) respectively. Note that 4 time steps were necessary in iteration 3, the result of the third iteration with 3 time steps is showed in (d). White regions are in state S_1 and S_2, blue regions are in state S_3, whereas red regions are in state S_4.

run the algorithm with 2 pursuers and 10 time steps in the first iteration, after 45 minutes, the algorithm had not finished the first iteration.

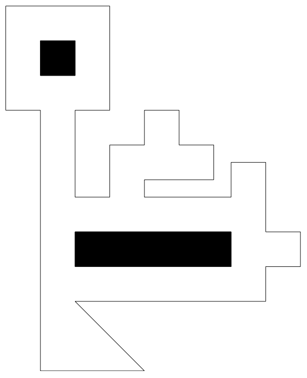

Fig. 7. A complex environment.

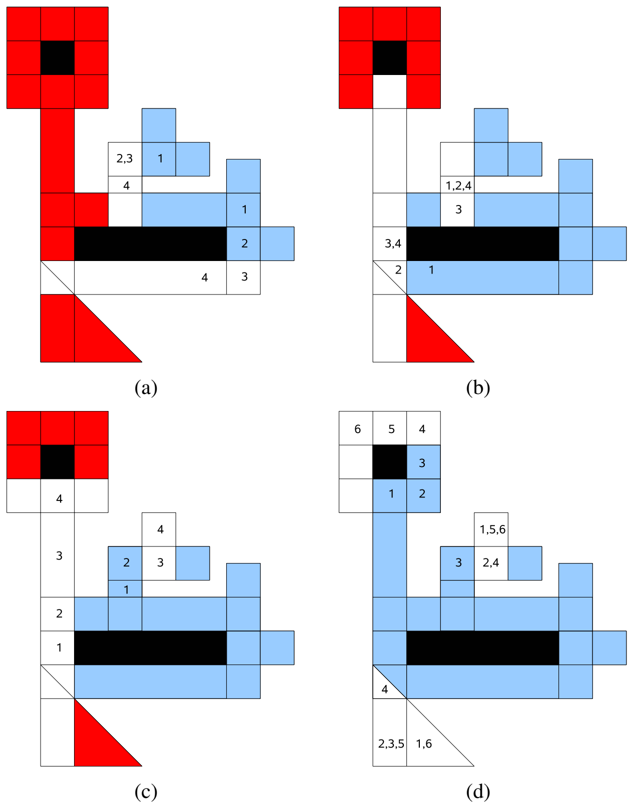

Fig. 8. The results of running the algorithm on the environment in Figure 7. The search problem is solved with 3 pursuers in four iterations where the results of iteration 1, 2, 3 and 4 are shown in (a), (b), (c) and (d) respectively. After iteration 3, no improvement is achieved with 2 pursuers, thus one more pursuer is added in iteration 4. The number of time steps was also increased. White regions are in state S_1 and S_2, blue regions are in state S_3, whereas red regions are in state S_4.

VI. Conclusions

In this paper a new approach for solving multi-pursuer pursuit evasion problems in polygonal environments has been proposed. In this approach a mixed integer linear programing (MILP) formulation was used in an iterative RHC fashion, to address the NP-hard pursuit problems.

From the simulations, conclusions can be drawn that the algorithm has the potential to drastically reduce computational cost, at the expense of possibly sub-optimal pursuer paths. For example, using the RHC approach a complex problem was solved in under 20 s, while the search for an optimal single iteration solution was aborted after running for 45 minutes.

References

[1] ILOG CPLEX. 10.2 User's Manual. *ILOG Inc., Gentilly, France*, 2007.

[2] B.P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, 25(4):299, 2006.

[3] L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 1999.

[4] G. Hollinger, S. Singh, and A. Kehagias. Efficient, Guaranteed Search with Mult-Agent Teams. *2009 Robotics: Science and Systems Conference, RSS*, 2009.

[5] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884, 2005.

[6] A. Kolling and S. Carpin. The GRAPH-CLEAR problem: definition, theoretical properties and its connections to multirobot aided surveillance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1003–1008, 2007.

[7] B. Mettler and E. Bachelder. Combining on-and offline optimization techniques for efficient autonomous vehicle's trajectory planning. In *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, CA, USA*, 2005.

[8] P. Ögren (Editor), D. Anisi, D. Berglund, D. Dimarogonas, H. Gustavsson, L. Hedlin, J. Hedström, X. Hu, K. H. Johansson, F. Katsilieris, V. Kaznov, P. Lif, M. Lindhé, U. Nilsson, M. Persson, M. Seeman, P. Svenmarck, and J. Thunberg. Results from the project aures: Autonomous ugv-system for reconnaissance and surveillance. Technical Report FOI-R-2783-SE, Swedish Defence Research Agency (FOI), 2009.

[9] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21:863, 1992.