Auton Robot
DOI 10.1007/s10514-011-9247-y

# A Mixed Integer Linear Programming approach to pursuit evasion problems with optional connectivity constraints

**Johan Thunberg · Petter Ögren**

Received: 12 October 2010 / Accepted: 16 July 2011


**Abstract** In this paper, we address the multi pursuer version of the pursuit evasion problem in polygonal environments. By discretizing the problem, and applying a Mixed Integer Linear Programming (MILP) framework, we are able to address problems requiring so-called recontamination and also impose additional constraints, such as connectivity between the pursuers. The proposed MILP formulation is less conservative than solutions based on graph discretizations of the environment, but still somewhat more conservative than the original underlying problem. It is well known that MILPs, as well as multi pursuer pursuit evasion problems, are NP-hard. Therefore we apply an iterative Receding Horizon Control (RHC) scheme where a number of smaller MILPs are solved over shorter planning horizons. The proposed approach is implemented in Matlab/Cplex and illustrated by a number of solved examples.

**Keywords** Pursuit evasion · MILP · Search

## 1 Introduction

The visibility based pursuit evasion problem addressed here was first proposed by Suzuki and Yamashita (1992) and later studied in e.g., (Guibas et al. 1999; Gerkey et al. 2006; Isler et al. 2005; Hollinger et al. 2009, 2010; LaValle and Hinrichsen 2002; Tovar and LaValle 2008; Simov et al. 2009; Yu and LaValle 2008; Kolling and Carpin 2007). 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 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. (1999), where it was also pointed out that the extension of that same approach presented considerable challenges in the multi-pursuer case. The two pursuer case was addressed in Simov et al. (2009), but since Guibas et al. also showed that the general problem is indeed NP-hard, solving problems with additional pursuers in reasonable time will be very hard. The concepts of Guibas et al. (1999) were built upon in Gerkey et al. (2006), where a field of view limitation was incorporated into the problem. The one-pursuer case was successfully treated, but once again, the multi-pursuer case turned out to be computationally intractable. Additional aspects of the problem have also been addressed, such as curved environments (LaValle and Hinrichsen 2002) and bounded speed evaders (Tovar and LaValle 2008).

As optimal deterministic strategies with guaranteed capture are hard to find, the option of using randomized strategies was explored in Isler et al. (2005). It was shown that a single pursuer can locate an evader in any simply con-

The first author would like to gratefully acknowledge the financial support by the Swedish National Space Technology Research Programme (NRFP).

J. Thunberg
Division of Optimization and Systems Theory, Department of Mathematics, Royal Institute of Technology (KTH), 100 44 Stockholm, Sweden
e-mail: johan.thunberg@math.kth.se

P. Ögren (✉)
Department of Aeronautical and Systems Technology, Swedish Defence Research Agency (FOI), 164 94 Stockholm, Sweden
e-mail: petter.ogren@foi.se

Published online: 18 August 2011

nected environment with high probability. Randomized approaches such as these are clearly an option for the multi-pursuer problem, but will not be investigated here. An interesting approach, focusing on the discrete time evolution of the cleared and contaminated parts of the environment was presented in Yu and LaValle (2008). The approach is related to the one presented here, but uses graphs, instead of a MILP, to keep track of the status of the different areas. Once the evader is found, the problem of how to move the pursuers in order to keep the evader in view arises. Such problems are sometimes referred to as target tracking problems (Jung and Sukhatme 2002) but will not be considered in this paper.

A problem that is closely related to the visibility based pursuit evasion problem is the one where the evader and pursuers are constrained to move in a graph. All instances of the former can be more or less conservatively discretized into the latter (Hollinger et al. 2010). One version of the graph search problem is called GRAPH-CLEAR, and was studied in Kolling and Carpin (2007). 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 that most polygonal environments can be divided into rooms and doors in many different ways. The rooms could either be very small, and trivially clearable by a single pursuer, or quite big, making the room clearing a non-trivial subproblem. The potential drawback of making the rooms very small is that the quality of the solution might be reduced since pursuers can not see from one room to another. Therefore, one could imagine a hierarchical approach with a global GRAPH-CLEAR problem and a polygonal pursuit evasion problem for each room. The latter can be solved by e.g., the approach presented here, to find how many pursuers are needed to clear each room.

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 Schouwenaars et al. (2001), Bellingham et al. (2002), where MILP was used to find detailed trajectories over a short planning horizon. The MILP computations were then iterated in an 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 Schouwenaars et al. (2001), Bellingham et al. (2002), 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 proposed approach can be seen as a lying in between the exact approaches and the graph based ones in the following sense. The visibility properties of the different areas is captured by the MILP in a way that is more conservative than the exact approaches (Guibas et al. 1999; Simov et al. 2009), but less conservative than the graph based ones (Hollinger et al. 2010).

This paper is an extension of the basic concepts presented in the conference paper (Thunberg and Ögren 2010). The new material includes decomposition (Sect. 5), connectivity constraints (Sect. 6) and results on problems requiring so-called recontamination (Sect. 7). The main contribution of the paper is the MILP formulation, including variations, 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 Sect. 2 the polygonal pursuit evasion problem is formally stated and in Sect. 3 the proposed solution is described. Then, Sect. 4 describes an RHC extension of the algorithm, Sect. 5 describes how larger problems can be addressed, and Sect. 6 describes how additional constraints, such as connectivity can be incorporated. Finally, Sect. 7 contains simulation examples to illustrate the approach and Sect. 8 concludes the paper.

## 2 Problem formulation

Following Guibas et al. (1999), 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 $e : [0, \infty) \to F$ 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 can not. Given $N$ pursuers, let $p_i(\tau) : [0, \infty) \to F$ denote the position of the $i$:th pursuer, and $P = \{p_1, \ldots, p_N\}$ 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 continuous function $e : [0, \infty) \to F$ there exists a time $\tau$ and an $i$ such that $e(\tau) \in V(p_i(\tau))$, i.e., the evader will always be seen by some pursuer, regardless of its path.*

It was shown in Guibas et al. (1999) 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 (Isler et al. 2005), 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 (Schouwenaars et al. 2001; Bellingham et al. 2002) and we will argue that they are applicable to Problem 1 as well.

## 3 Proposed solution

In the proposed solution, we first discretize Problem 1 by partitioning the polygonal free space $F$ into a set of *convex* regions, $F = \cup_{i \in J} F_i$, $J = \{1, \ldots, K\}$. 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.

### 3.1 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 Fig. 1. 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 at most $O(n^2)$ 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 $k \in O(n^2)$. □

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 Fig. 2(a).

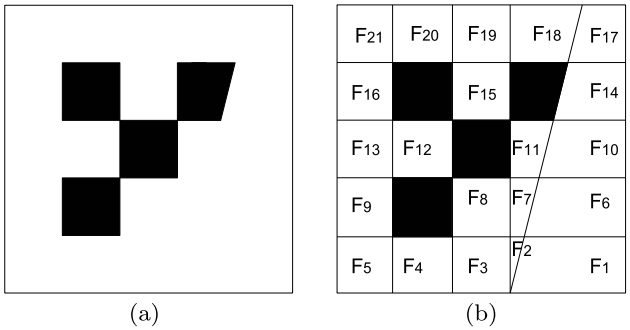


**Fig. 1** An example environment with one irregularly shaped obstacle (**a**), and the corresponding partition of the free space $F$ into convex polygons $F_1 \ldots F_{21}$ (**b**)

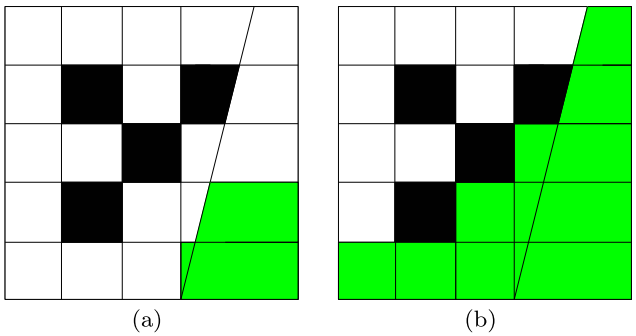


**Fig. 2** The neighborhood that can be moved to, $M_1$ (**a**) and the neighborhood that can be seen, $N_1$ (**b**), from area $F_1$

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 considered not visible at all. In some approaches, such as (Guibas et al. 1999; Gerkey et al. 2006), this is not the case, but in others, such as the graph search approaches (Hollinger et al. 2010), an even more conservative discretization is needed to address open areas.

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

### 3.2 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 $t \in \{1, 2, \ldots, T\}$. 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 $\{i : \theta_{it} = 1\}$.

**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*

*Constraints addressing*: $\lambda_{it}$ (*pursuer locations*),

$$\sum_{i \in J} \lambda_{it} - N = 0, \tag{2}$$

$$N - (N - 1)\lambda_{it} - \sum_{j \in M_i} \lambda_{jt} \geq 0, \tag{3}$$

$$\sum_{j \in M_i} \lambda_{jt} - \lambda_{i(t-1)} \geq 0, \tag{4}$$

$$2 - \sum_{j \in M_i} \lambda_{j(t-1)} - \lambda_{it} \geq 0. \tag{5}$$

*Constraints addressing*: $\sigma_{it}$ (*seen regions*)

$$\sum_{j \in N_i} \lambda_{jt} - \sigma_{it} \geq 0, \tag{6}$$

$$\sigma_{it} - \lambda_{jt} \geq 0, \quad \forall j \in N_i \tag{7}$$

*Constraints addressing*: $\theta_{it}$ (*unseen cleared regions*)

$$\sigma_{jt} + \theta_{jt} - \theta_{it} \geq 0, \quad \forall j \in M_i - \{i\}, \tag{8}$$

$$\sigma_{i(t-1)} + \theta_{i(t-1)} - \theta_{it} \geq 0, \tag{9}$$

$$1 - \sigma_{it} - \theta_{it} \geq 0, \tag{10}$$

$$\theta_{i1} = 0, \tag{11}$$

*where $\alpha \in [0, 1]$, $i \in J$ and $t \in \{2, 3, \ldots, T\}$ in (4), (5) and (9) and $t \in \{1, 2, \ldots, T\}$ in the other constraints.*

We will now motivate all the constraints and then make remarks about the objective function and the fact that the constraints are somewhat conservative. In (2) we make sure that there are exactly $N$ pursuers at each time $t$. Constraints (3)–(5) make sure, in a somewhat conservative way, that the pursuers move between neighboring regions one step at a time. Constraint (3) states that if a region is occupied, then all its neighbors must be unoccupied. (4) Makes sure that if a region was occupied in the last time step, at least one of its neighbors must be occupied in this time step. (5) Implies that if a region is occupied, at most one of its neighbors was occupied the last time step.

Moving on to the constraints making sure that the right regions are labeled as seen, we note that (6) makes sure that if there are no pursuers in the $N_i$ neighborhood, then the region can not be seen, and (7) makes sure that if there is a pursuer in one of the $N_i$ neighborhood regions, then it must be seen.

Finally, looking at the constraints governing the unseen but cleared regions, we note that (8) implies that if any neighbor of a region is not seen nor cleared, then the region can not be cleared itself. (9) Implies that a cleared region must have been either cleared or seen in the previous time step. (10) Then states that a region can not be seen and cleared but unseen at the same time. (11) Finally makes sure that there are no cleared areas at starting time. Note also that the objective function makes sure that regions that can be marked as cleared are marked as cleared.

*Remark 2* (Objective function) 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 Sect. 7 below we will see that $\alpha = 1$ is actually the best measure of progress for the clearing task. By removing the visible region from the objective function, we decrease the risk of the pursuers staying in locations where they see a large area, instead of moving on to clear additional areas.

*Remark 3* (Conservative constraints) Note also that constraints (3)–(5) implies that pursuers never occupy the same or neighboring regions. This restriction is somewhat conservative, as it is not present in Problem 1. A less conservative version of the proposed approach can be created at the price of increased complexity. This is done by adding one more index $k$ on the variable $\lambda$, i.e. $\lambda_{itk}$, where $k$ is the pursuer index, and letting $\lambda_{ijk} = 1$ if and only if pursuer $k$ is in region $i$ at time $t$. In this representation however the number of variables and constraints grow linearly with the number of pursuers, compared to the presented representation in which they are constant.

**Lemma 2** *A feasible solution to Problem 2 can be used to generate pursuer paths $p_i(\tau), \tau \in [0, T'], i \in \{1, \ldots, N\}$, 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 $S_4$ 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 (2) it is guaranteed that there are exactly $N$ pursuers at each time $t$. Constraints (2)–(5) together guarantee that a pursuer move between adjacent regions in consecutive time steps. Now, pursuer paths $p_i(\tau)$ can be created from $\lambda_{it}$ where all pursuers cross borders between the $F_i$ at the same time. Finally, a mapping between continuous time $\tau$ 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 (6) and (7) the variable $\sigma_{it}$ 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 (8) 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 (9) 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 (10) 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 (11) verifies 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.,*

$$\sum_{i \in J} (\sigma_{iT} + \theta_{iT}) = card(J), \tag{12}$$

*is a solution to Problem 1.*

*Proof* A straightforward application of Lemma 2 above. □

*Remark 4* (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 5* (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.

## 4 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.

### 4.1 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 Sect. 1 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 and for all 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 $\sigma_{iT}$ and $\theta_{iT}$ satisfies*

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

   *the whole area is cleared, and the algorithm terminates.*
3. *Else, if there was no increase in $\sum_{i \in J} (\sigma_{iT} + \theta_{iT})$*
   (a) *If more pursuers are available, use them, i.e. increase the number of pursuers $N$,*
   (b) *else, if computational resources allow it, increase the horizon length $T$,*

(c) *else, decompose the problem into smaller subproblems, as described in Algorithm 2.*

4. *Prepare a new RHC iteration by removing constraint* (11) *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.

**Lemma 4** *If Algorithm 1 terminates, a solution to Problem 1 is found.*

*Proof* A straightforward application of Lemma 3 above. □

### 4.2 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* $\sigma_{it}$ *and* $\theta_{it}$ *in Problem 2 are relaxed such that they are no longer binary variables but belong to* $[0, 1]$, *i.e.*

$$0 \leq \sigma_{it} \leq 1, \tag{14}$$

$$0 \leq \theta_{it} \leq 1. \tag{15}$$

Using CPLEX 10.2, Problem 3 seems to be between 2 and 20 times as fast as the original formulation.

**Lemma 5** *The pursuer paths* $\lambda_{it}$ *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 (7), also if $\lambda_{jt} = 0, \forall j \in N_i$ then $\sigma_{it} = 0$ by (6), thus $\sigma_{it}$ is binary. Now let $(\lambda, \sigma, \theta^b)$ be a (possibly suboptimal) solution to Problem 2 in which only $\theta^b$ differs from the solution of Problem 3. Let $Z_2$ and $Z_3$ be the cost of the solutions to Problems 2 and 3 respectively. By (8), (9) and (10) we get that for each $\theta_{it} \in (0, 1]$, $\theta^b_{it} = 1$. This implies that $Z_3 \leq Z_2$, but problem 3 is a relaxation of problem 2 which implies $Z_2 \leq Z_3$. Thus $Z_2 = Z_3$. □

## 5 Decomposition of large environments

As noted in Sect. 1, one way of reducing the complexity of large problem instances is to use a hierarchical decomposition with a GRAPH-CLEAR problem at the top and instances of Problem 1 at the lower level. Another option is to decrease the problem size and complexity by placing stationary pursuers at positions where they cover large areas, and then solve instances of Problem 1 in the remaining unseen parts of $F$. This approach is described below.

**Algorithm 2**

1. *Solve the MILP with k pursuers and one time step with* $\alpha = 0$. *This corresponds to maximizing the number of seen polygons by k pursuers, i.e., solving an Art Gallery problem* (Urrutia 2000).
2. *Remove all polygons that are in state* $S_1$ *or* $S_2$ *from* $F$, *leaving a possibly disconnected* $F$.
3. *Apply Algorithm 1 to each connected component of* $F$ *and let* $q$ *be the maximal number of pursuers needed.*
4. *The number of pursuers needed to clear the original* $F$ *is now* $k + q$.

## 6 Connectivity constrained search

An area receiving an increasing amount of interest is communication aware motion planning (Anisi et al. 2010; Lindhe and Johansson 2008). In this section we will show how the proposed MILP framework can be extended to handle one such problem, namely the problem where the whole group of pursuers shall be connected in a line-of-sight graph at a given time instant $t'$. We will present two sets of constraints. The first set corresponds to a general line-of-sight graph, while the second set corresponds to a special case, a star shaped line-of-sight graph, where one of the pursuers sees all others.

For the general case we extend Problem 2 with a set of binary variables $u_{ij}$, where $i \in J$ and $j \in \{1, 2, \ldots, N\} = J_P$. Below we will first add constraints (16–18) to ensure that $u_{ij} = 1$ if and only if pursuer $j$ is located in region $i$ at time $t'$, constraints . We then add constraint (19) so that all pursuers, except pursuer number one, is visible to a pursuer with an index that is lower than its own. The list of constraints is as follows.

$$\lambda_{it'} - u_{ij} \geq 0, \quad \forall i \in J, \ j \in J_P, \tag{16}$$

$$\sum_{j \in J_p} u_{ij} \leq 1, \quad \forall i \in J, \tag{17}$$

$$\sum_{i \in J} u_{ij} = 1, \quad \forall j \in J_P, \tag{18}$$

$$\sum_{l=1}^{j-1} \sum_{k \in N_i} u_{kl} - u_{ij} \geq 0, \quad \forall i \in J, \ j \in J_P - \{1\}. \tag{19}$$

Equation (16) states that $u_{ij}$ can be equal to 1 only if $\lambda_{it'}$ is equal to 1. Equations (17–18) together guarantee that there is one and only one unique $u_{ij} = 1$ for each $\lambda_{it'} = 1$. Equation (19) states that $u_{ij}$ where $i \in J$ and $j \in \{2, 3, \ldots, N\}$ can only be equal to 1 if there is at least one pair $(k, l)$, where $k \in N_i$ and $l \in \{1, 2, \ldots, j-1\}$, such that $u_{k,l} = 1$. Given (16–18), equation (19) guarantees the existence of a line-of-sight graph.

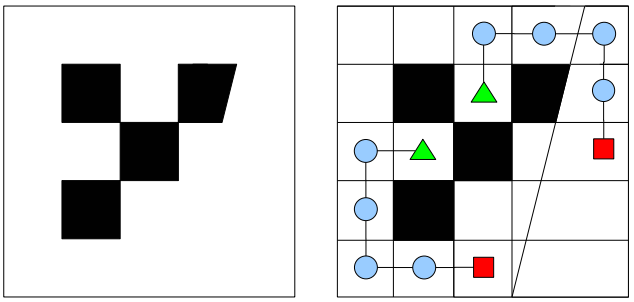


**Fig. 3** The results of running Algorithm 1 with a 6 step planning horizon (**b**) in the environment in (**a**). A *green triangle* denotes the start of a pursuer path, and a *red square* denotes the stop of a pursuer path, *blue discs* denote intermediate steps

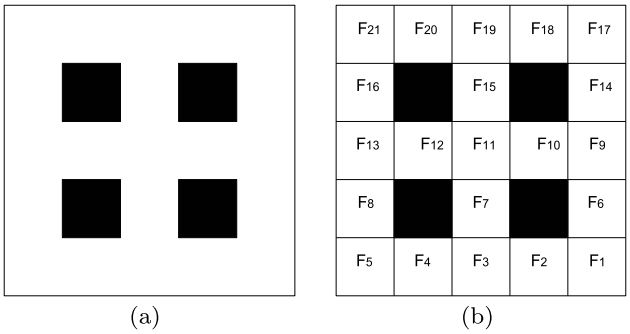


**Fig. 4** The partition (**b**) of a Manhattan grid with four obstacles (**a**)

*Remark 6* If these constraints are used at all time steps, the entire search is performed with the group of pursuers connected in a line-of-sight graph.

An alternative to the general case above, is the problem of creating a line-of-sight graph where one single vertex is connected to all others. This results in a smaller set of binary variables $u_i$. The topology of the connected visibility graph is defined by the first constraints below, whereas the last constraint, with the sum of all $u_j$:s, implies that there must exist such a graph.

$$\lambda_{it'} + \sum_{j \in N_i} \lambda_{jt'} - (N+1)u_i \geq 0, \quad i \in J, \tag{20}$$

$$\sum_{j \in J} u_j \geq 1. \tag{21}$$

Both these sets of constraints will result in a connected graph independently of the number of pursuers.

## 7 Simulation examples

When running Algorithm 1, it turns out that the best results are found using $\alpha = 1$. A drawback of the more intuitive

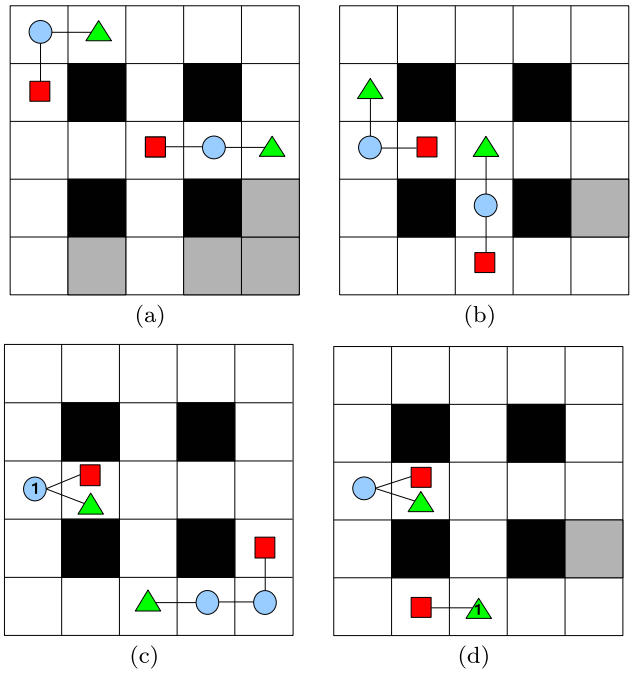


**Fig. 5** The results of running the algorithm on a Manhattan grid with four obstacles. The partition is showed in Fig. 4, and the search problem is solved with 2 pursuers in three iterations where the results of iteration 1, 2 and 3 are shown in (**a**), (**b**) and (**c**) respectively. Note that 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 states $S_1$, $S_2$ or $S_3$, whereas grey regions are in state $S_4$. A *green triangle* denotes the start of a pursuer path, and a *red square* denotes the stop of a pursuer path, *blue discs* denote intermediate steps. A number $i$ inside a triangle, disc or square indicates that the pursuer waits an additional $i$ time steps in the region

$\alpha = 1/2$ 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, 3.00 GHz with 4 cores, running the MILP software CPLEX 10.2 (CPLEX 2007). Furthermore, all results were found using the relaxed version, Problem 3, as it was found to be between 2 and 20 times as fast as the non-relaxed formulation. Finally, we note that the proposed approach is not directly suitable for very large problems. Such problems can be decomposed into smaller ones, either by applying Algorithm 2 or using a GRAPH-CLEAR formulation, as described in Sect. 1. The performance and limitations of the approach can be seen from the examples below.

The first problem instance is depicted in Fig. 3(a) with the corresponding solution in Fig. 3(b). With a time horizon of $T = 6$, and $\alpha = 1/2$, a single RHC iteration was needed, and found in 4 seconds.

The second problem instance is shown in Fig. 4, with corresponding solution in Fig. 5(a–c). The problem was first solved in 3 RHC iterations using a total number 10 time

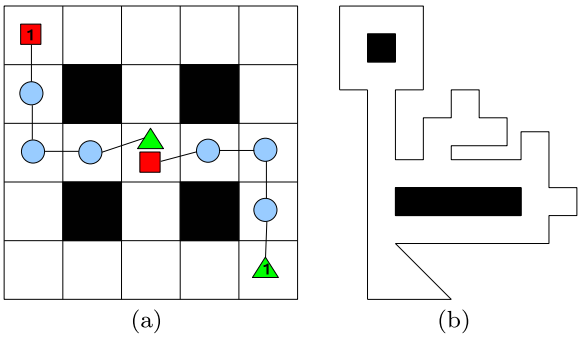


**Fig. 6** (**a**) The solution of running the algorithm on the environment in Fig. 4 with 2 pursuers for 6 time steps. The *color coding* of the regions are as in Fig. 5. (**b**) A complex environment with fewer loops

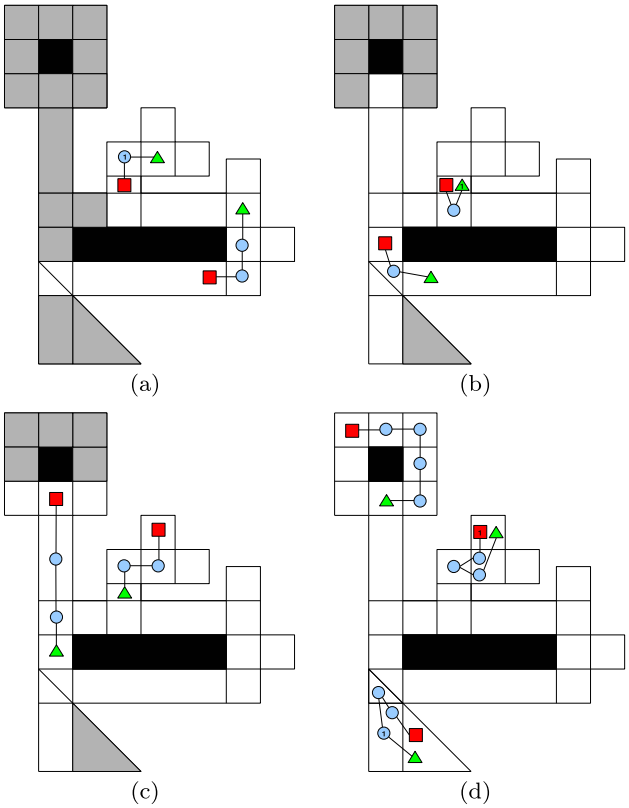


**Fig. 7** The results of running the algorithm on the environment in Fig. 6(b). The search problem is solved with 3 pursuers in four iterations where the results of iteration 1, 2, 3 and 4 are shown in (**a**), (**b**), (**c**) and (**d**) respectively. After iteration 3, no improvement is achieved with 2 pursuers, thus one more pursuer is added in iteration 4. The number of time steps was also increased. The *color coding* of the regions are as in Fig. 5

steps. The computational time was about 3 seconds for each iteration resulting in a computational time of 9 seconds in total. 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 5(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 computational time of 110 seconds, see Fig. 6.

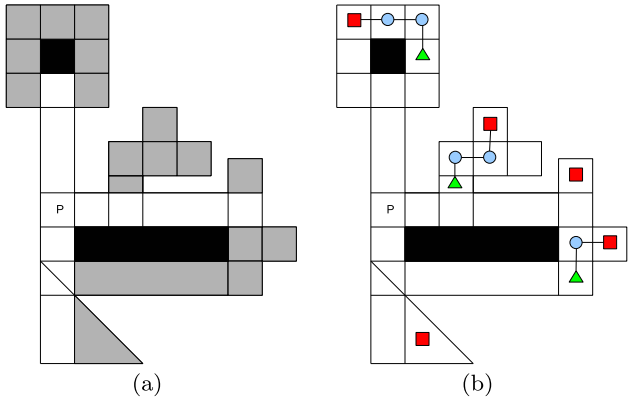


**Fig. 8** The results or running Algorithm 2 with two pursuers on the environment in Fig. 6(b). In step 1, we use $k = 1$, i.e., one pursuer is used to cover as much of the area as possible, and the resulting position is shown in (**a**). Then the remaining five connected components of $F$ are searched one after the other with one pursuer, (**b**). The *color coding* of the regions are as in Fig. 5

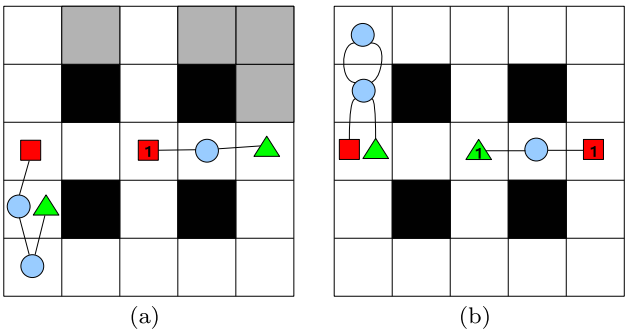


**Fig. 9** The result of running the algorithm on the Manhattan grid with the additional connectivity constraints (20) and (21) active at the final time of each iteration. The problem was solved in two iterations using 4 and 5 timesteps respectively, and one can verify that the pursuers can see each other at the final time of each iteration

The third problem instance is shown in Fig. 6(b) with corresponding solution in Fig. 7. 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 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.

The solution of the fourth problem is shown in Fig. 8. The environment is the same as the third problem, but the solution is found using decomposition, i.e., Algorithm 2, as presented in Sect. 5. Step 1 of the algorithm was solved in 2 seconds with $k = 1$, i.e. one stationary pursuer was used. The position of this pursuer is shown in Fig. 8(a). The search

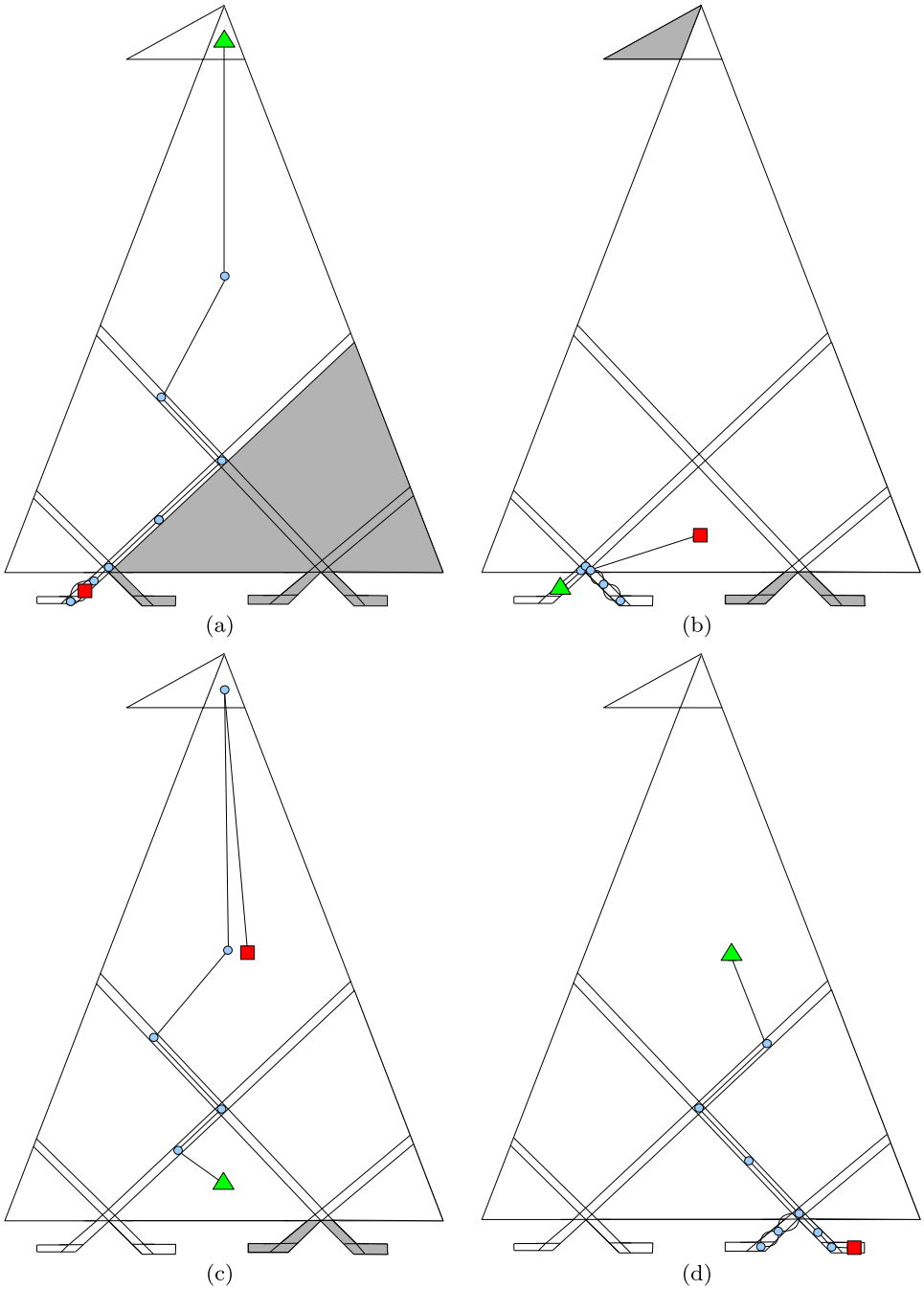


**Fig. 10** The results of running the algorithm on a problem requiring recontamination. Note that the upper part of the area is first cleared, then contaminated and finally cleared once again, and that recontamination is necessary when clearing the whole area with one pursuer

of the remaining unseen parts of $F$, in this case five disconnected regions, took 2 seconds each and is shown in Fig. 8(b). Thus the problem was solved in a total of 12 seconds, using 2 pursuers.

The fifth problem illustrates the connectivity constraints. The Manhattan grid in Fig. 4 was solved with the algorithm in two iterations with 4 and 5 time steps respectively, see Fig. 9. The additional connectivity constraints (20) and (21) were active at the final time of each iteration. In the figure one can see that the two pursuers are indeed connected by a free line of sight at the final time of each iteration.

The sixth problem is a single pursuer one, solving a problem that requires so-called recontamination, Fig. 10. The problem is taken from Guibas et al. (1999), where it was shown that some problems require a linear (in edges) number of recontaminations, i.e., some areas need to change back and forth between being contaminated and cleared a number of times, before finally being cleared. In this prob-

**Table 1** The computational time in seconds as a function of environment size ($E_1$ to $E_4$) and number of pursuers. Below, '–' denotes that starting positions with the required separation could not be found

| Pursuers | $E_1$ | $E_2$ | $E_3$ | $E_4$ |
|---|---|---|---|---|
| 2 | $<1$ | 7 | $>1700$ | $>1700$ |
| 3 | $<1$ | 26 | $>1700$ | $>1700$ |
| 4 | $<1$ | $<1$ | $>1700$ | $>1700$ |
| 5 | $<1$ | 8 | $>1700$ | $>1700$ |
| 6 | $<1$ | $<1$ | $>1700$ | $>1700$ |
| 7 | $<1$ | 13 | $>1700$ | $>1700$ |
| 8 | $<1$ | $<1$ | $>1700$ | $>1700$ |
| 9 | $<1$ | 13 | $>1700$ | $>1700$ |
| 10 | $<1$ | $<1$ | 21 | $>1700$ |
| 11 | – | $<1$ | 2 | $>1700$ |
| 12 | – | $<1$ | 39 | $>1700$ |
| 13 | – | 72 | 38 | $<1$ |
| 14 | – | $<1$ | $<1$ | 18 |
| 15 | – | – | $<1$ | 10 |

lem, the recontaminated area is at the very top, and the recontamination is shown in Fig. 10b.

In the seventh problem, we investigate how the computational time varies with the size of the environment to be cleared, and the number of pursuers. Four different search environments of increasing size were used. These were created as follows from the four subsets $A, B, C$, and $D$ shown in Fig. 11. Environment $E_1 = A$, $E_2 = A \cup B$, $E_3 = A \cup B \cup C$, and $E_4 = A \cup B \cup C \cup D$. We run Algorithm 1 with $\alpha = 1$ and time horizon $T = 3$ on environments $E_1$ to $E_4$ with the number of pursuers ranging from 2 to 15. The resulting computational times can be found in Table 1, where an '–' indicate that feasible starting positions satisfying (3)–(5) could not be found. As expected, we can see that the computational time increases with the environment size. However, it does not grow with the number of pursuers and solutions to the more complex environments $E_3$, $E_4$ can only be found, within the given time limits, with a larger number of pursuers.

## 8 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 programming (MILP) formulation was used in an iterative RHC fashion to address NP-hard pursuit evasion problems.

It was shown that the proposed approach can solve problems requiring so-called recontamination, if the planning horizon is long enough to capture the benefits of this recontamination. We have also seen that the approach is flexible, allowing additional aspects, such as connectivity constraints, to be included in the formalism. Finally, the approach represents a middle ground between the computationally efficient but conservative graph approaches and the non-conservative but computationally challenging exact solutions.

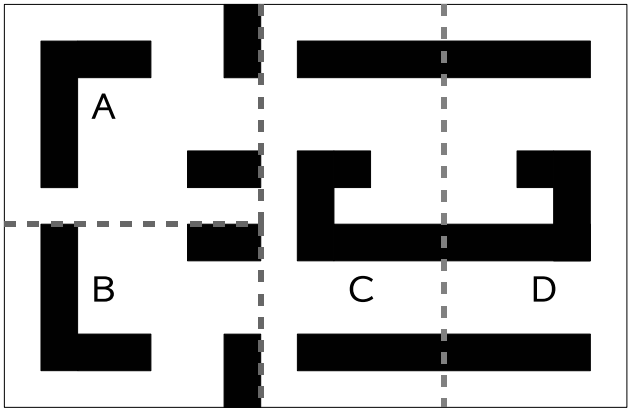


**Fig. 11** The environments used to investigate computational time as a function of problem size and the number of pursuers. The four environments of Table 1 are $E_1 = A$, $E_2 = A \cup B$, $E_3 = A \cup B \cup C$, and $E_4 = A \cup B \cup C \cup D$

**Acknowledgements** We would like to thank Erik Peldan and Richard Hermanson for their help in running the simulations reported in Table 1.

## References

Anisi, D., Ögren, P., & Hu, X. (2010). Cooperative minimum time surveillance with multiple ground vehicles. *IEEE Transactions on Automatic Control*, *55*. doi:10.1109/TAC.2010.2047438.

Bellingham, J., Richards, A., & How, J. (2002). Receding horizon control of autonomous aerial vehicles. In *American Control Conference: Vol. 5* (pp. 3741–3746). New York: IEEE Press.

CPLEX, I. (2007). *10.2 user's manual*. ILOG Inc., Gentilly.

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

Guibas, L., Latombe, J., LaValle, S., Lin, D., & Motwani, R. (1999). A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, *9*, 471–493.

Hollinger, G., Singh, S., & Kehagias, A. (2009). Efficient, guaranteed search with multi-agent teams. In *2009 Robotics: science and systems conference, RSS*

Hollinger, G., Singh, S., & Kehagias, A. (2010). Improving the efficiency of clearing with multi-agent teams. *The International Journal of Robotics Research*, *29*(8), 1088–1105.

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

Jung, B., & Sukhatme, G. (2002). Tracking targets using multiple robots: the effect of environment occlusion. *Autonomous Robots*, *13*(3), 191–205.

Kolling, A., & Carpin, S. (2007). 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* (pp. 1003–1008).

LaValle, S., & Hinrichsen, J. (2002). Visibility-based pursuit-evasion: The case of curved environments. *IEEE Transactions on Robotics and Automation*, *17*(2), 196–202.

Lindhe, M., & Johansson, K. (2008). Communication-aware trajectory tracking. In *IEEE international conference on robotics and automation, ICRA* (pp. 1519–1524).

Schouwenaars, T., De Moor, B., Feron, E., & How, J. (2001). Mixed integer programming for multi-vehicle path planning. In *European control conference* (pp. 2603–2608).

Simov, B., Slutzki, G., & LaValle, S. (2009). Clearing a polygon with two 1-searchers. *International Journal of Computational Geometry and Applications*, *19*(1), 59–92.

Suzuki, I., & Yamashita, M. (1992). Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, *21*, 863.

Thunberg, J., & Ögren, P. (2010). An iterative Mixed Integer Linear Programming approach to pursuit evasion problems in polygonal environments. In *IEEE international conference on robotics and automation (ICRA)* (pp. 5498–5503). New York: IEEE Press.

Tovar, B., & LaValle, S. (2008). Visibility-based pursuit-evasion with bounded speed. In *Algorithmic foundation of robotics VII* (pp. 475–489). Berlin: Springer.

Urrutia, J. (2000). Art gallery and illumination problems. In *Handbook of computational geometry* (pp. 973–1027).

Yu, J., & LaValle, S. (2008). Tracking hidden agents through shadow information spaces. In *IEEE international conference on IEEE robotics and automation, ICRA 2008* (pp. 2331–2338). New York: IEEE Press.

**Johan Thunberg** was born in Stockholm, in 1982. He has received a Master of Science degree in Engineering Physics from the Royal Institute of Technology (KTH) in Stockholm, Sweden. After finishing his Master he has worked as a research assistant at the Swedish Defence Research agency (FOI), and at ENEA, a Swedish software company specialized in embedded systems.
Currently he is a Ph.D. student within the Centre for Autonomous Systems (CAS) and is funded by the Swedish Space Corporation (SSC) and the Swedish National Space Technology Research Programme (NRFP). His supervisior is Professor Xiaoming Hu. His research interests include vision based control and estimation, consensus problems, and optimization based Multi-agent coordination.

**Petter Ögren** was born in Stockholm, Sweden, in 1974. He received the M.S. degree in engineering physics and the Ph.D. degree in applied mathematics from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 1998 and 2003, respectively. In the fall of 2001, he visited the Mechanical Engineering Department, Princeton University, Princeton, NJ. He is currently working as a Senior Scientist with the Swedish Defence Research Agency (FOI), Stockholm, Sweden. His research interests include multirobot systems, formations, navigation, and obstacle avoidance.